1994年3月1日発行（毎月1回1日発行）第13巻3号通巻143号 昭和58年11月2日第三種郵便物認可

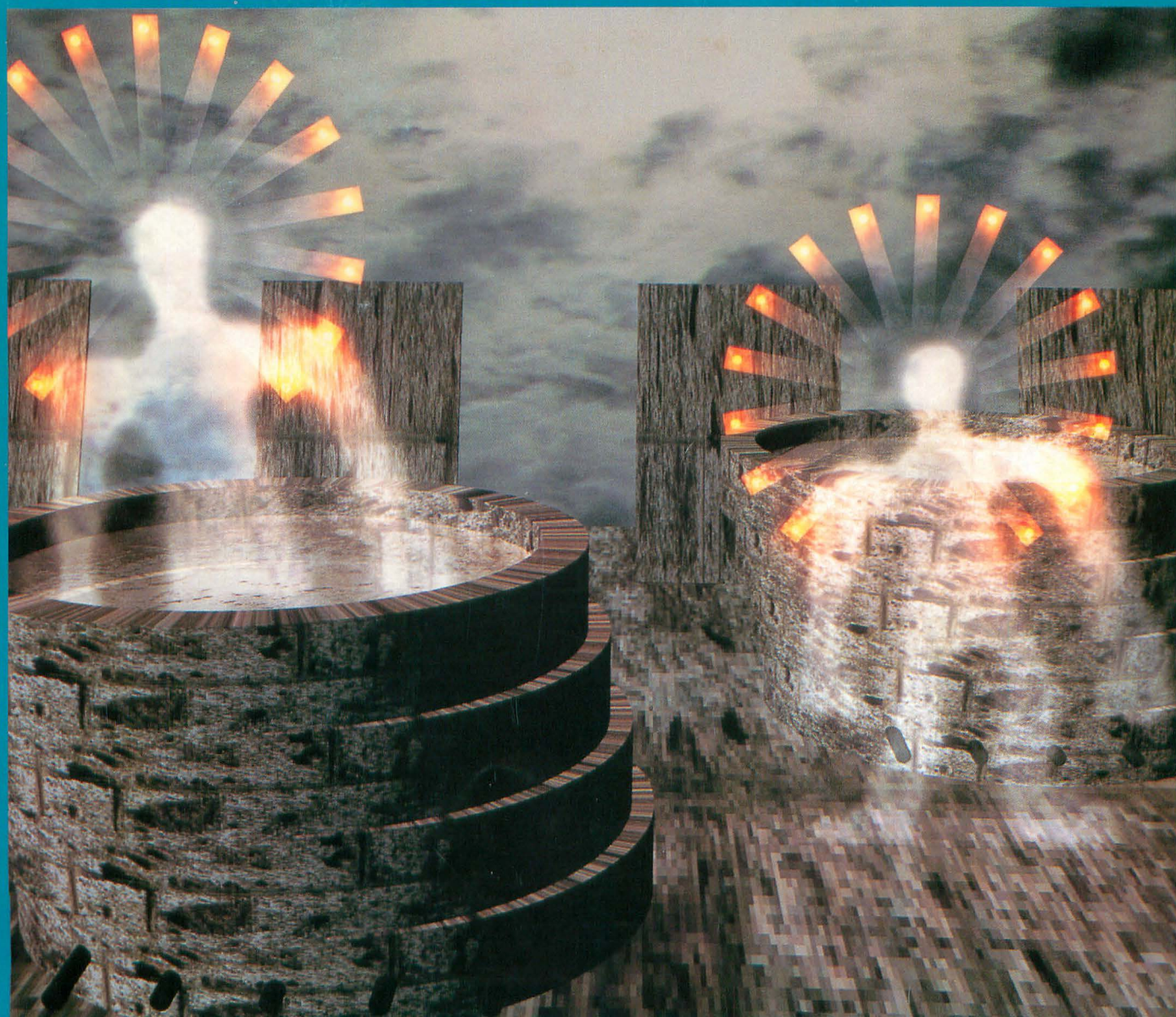
PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

特別付録5"2HD ひなまつりPRO-68K

SX-BASIC/Z's-EX/MATIER-EX/AMI/Morph!

新連載 石の言葉, 言葉の夢

新製品紹介 ビデオPC/Xsimm10

1994

SOFT BANK オー／エックス

特別定価800円

SHARP
目の付けどころが、
シャープでしょ。
夢の、頂きへ。
68ワールドの最高峰。
X68030
32bit PERSONAL WORKSTATION

演算速度4.3倍(当社10MHz機比)/2.4倍(当社XVI比)※1、動画ウィンドウに見る新創造次元。
選ばれた人だけが持つ感性によってX68030の扉はひらかれる。

X68000シリーズとして初の32ビットMPU MC68EC030を搭載して高速化を実現。

データキャッシュ、プログラムキャッシュをそれぞれ256バイト搭載したクロック周波数25MHzの高速32ビットMPUを搭載。演算速度は2倍以上(当社従来比)※1の高速化を実現しました。また数値演算プロセッサMC68882※2(25MHz)もサポート。大量の実数演算を必要とするクリエイティブワークやGUI環境の操作性など、実行速度の飛躍的な向上が図られています。(当社従来比)

※1 Dhrystn(四則演算)比。25MHz・データキャッシュオン・プログラムキャッシュオンでMC68000/10MHz時の約4.3倍、16MHz時の約2.4倍。

※2 数値演算プロセッサCZ-5MP1標準価格54,800円(税別):本体内の専用ソケットに取りつけ可能。

65,536色表示、動画表示を実現。さらにパワーアップしたSX-WINDOWver.3.0。

X68000独自のウィンドウシステムとして定評の「SX-WINDOWver.2.0」をさらに強化した「SX-WINDOWver.3.0」を標準装備。新たに、65,536色の自然色グラフィック表示を可能とした『グラフィックウィンドウ』※を搭載。またアニメーション動画をウィンドウ上で表現でき、手軽にコンピュータアニメーションが楽しめる『CGAウィンドウ』、さらに従来のエディタのイメージを一新、高度な日本語文書作成をサポートするSX-WINDOW対応の高機能日本語マルチフォントエディタを標準装備。アウトラインフォントの展開もさらに高速化が図られています。

※SX-WINDOW上の512×512ドットのエリア内で表示可能。

GUIに対応する大容量メインメモリを搭載。

メインメモリは標準で4Mバイト、複数のアプリケーションをウィンドウ上で同時に使用するなど大量のデータ処理に対応。また本体内の増設で、I/Oスロットを使用せず最大12Mバイトまで拡張できます。拡張したメモリはすべて32ビットバスによる高速アクセスが可能、優れた拡張環境でシステムパワーアップをサポートします。

※メモリ増設には、4MB内部増設RAMボードCZ-5BE4標準価格54,800円(税別)、4MB増設RAMモジュールCZ-5ME4標準価格49,800円(税別)をご使用ください。なおCZ-5ME4はCZ-5BE4上に装着します。

X68000シリーズの高機能を継承した上で、さらに使いやすさの向上を図ったコンパチビリティ重視設計※1、すぐに使える高機能ソフトを標準装備。

●25MHzでは速すぎるアプリケーションも、従来のクロック周波数(10MHz/16MHz)で動作可能なソフトコンパチ重視設計●65,536色同時発色の自然色グラフィックス(最大表示エリア512×512ドット)、1024×1024ドットの実画面エリアを持つ高解像度表示能力(最大表示エリア768×512ドット)、疑似高解像度スーパーインポーズ(インターレース方式/512×480ドット・専用ディスプレイテレビ使用時)を装備した高精細度自然色グラフィックス機能。●外部MIDI音源もコントロール可能※2、ウィンドウ上で手軽にコンピュータミュージックが楽しめるMIDI音源対応デバイスドライバ搭載●ステレオ8オクターブ8重和音FM音源、ADPCM搭載●プリンタ、RS-232C、SCSI、オーディオ入出力、イメージ入力など多彩なインターフェイスを装備。●日本語変換効率や操作性を高めた日本語フロントプロセッサASK68Kver3.0搭載。●従来のエディタのイメージを一新したSX-WINDOW対応の高速多機能日本語マルチフォントエディタ標準装備●日本語マルチフォントエディタ中に貼り付ける絵やグラフなどが簡単に作成できるグラフィックパターンエディタ●MIDI対応のX-BASIC。

※1 アプリケーションソフトおよび周辺機器のうち、一部動作しないものがあります。詳しくはシャープお客様相談窓口にお問い合わせください。

※2 別売のMIDIインターフェイスが必要です。

EXEクラブって何だ?

X68030/X68000を手に入れたら、やっぱり他のユーザーがどんな風に使っているのか気になるもの。ということでEXEクラブは、そんなあなたのための、他の68ユーザーとのコミュニケーションをバックアップする、情報交換の場です。

本体同梱の入会申込ハガキを送るだけで、自動的に無料入会。さらに下記の特典付き。

メリット1 会員ナンバー入りオリジナル会員電卓がもらえる。

メリット2 各種フェアご優待・イベント案内等、数々の特典がある。

130mmFD(5.25型) マンハッタンシェイプシリーズ

■X68000伝統のマンハッタンシェイプを継承 ■130㎜FDD(5.25型)2基搭載 ■80MBハードディスク内蔵(CZ-510C)※ ■マウス・トラックボール標準装備 ■ASCII準拠フルキーボード採用

※CZ-500Cには、80MB内蔵用ハードディスクドライブCZ-5H08/160MB内蔵用ハードディスクドライブCZ-5H16を用意しています。

X68030
32bit PERSONAL WORKSTATION

本体+キーボード+マウス・トラックボール
130㎜(5.25型)FDタイプ
CZ-500C-B(チタンブラック)標準価格398,000円(税別)
HD内蔵 CZ-510C-B(チタンブラック)標準価格488,000円(税別)
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

90mmFD(3.5型) コンパクトシリーズ

■32ビットのハイパワーを凝縮したコンパクトフォルム ■2DD対応90㎜FDD(3.5型)2基搭載 ■80MBハードディスク内蔵(CZ-310C)※ ■マウス標準装備 ■コンパクトキーボード採用

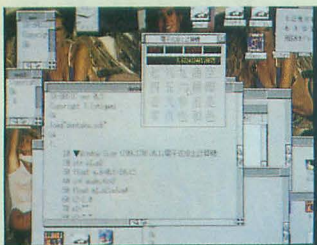
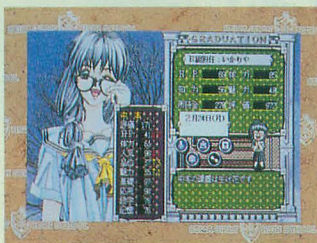
※CZ-300Cには、80MB内蔵用ハードディスクドライブCZ-5H08/160MB内蔵用ハードディスクドライブCZ-5H16を用意しています。

Compact

本体+キーボード+マウス
90㎜(3.5型)FDタイプ
CZ-300C-B(チタンブラック)標準価格388,000円(税別)
HD内蔵 CZ-310C-B(チタンブラック)標準価格478,000円(税別)
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

●消費税及び配送・設置・付帯工事費、使用済み商品の引き取り費等は、標準価格には含まれておりません。

■お問い合わせは… シャープ株式会社 コンシューマーセンター西日本相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 X68030 Oh!X 3係

特別企画　ひなまつりPRO-68K

卒業～GRADUATION

B-FIELD!

ビデオPC

Xsimm10

"実戦!"ゲーム作りのKNOW HOW

Oh!X

CONT

〈スタッフ〉
●編集長／前田　徹　●副編集長／植木章夫　●編集／山田純二　豊浦史子　高橋恒行　●協力／有田隆也　中森　章　林　一樹　吉田幸一　華門真人　吉田賢司　朝倉祐二　大和　哲　村田敏幸　丹　明彦　三沢和彦　長沢淳博　司馬　護　清瀬栄介　石上達也　柴田　淳　瀧　康史　横内威至　進藤慶到　●カメラ／杉山和美　●イラスト／山田晴久　江口響子　高橋哲史　川原由唯　●アートディレクター／島村勝頼　●レイアウト／元木昌子　ADGREEN　●校正／グループごじら

表紙絵：塚田　哲也

1994 MAR. 3

E N T S

■広告目次

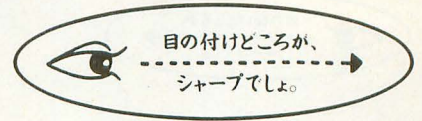
先が、面白くなる。

ウィンドウ環境のプラットホームを確立、SX-WINDOW ver.3.0

■実画面：1,024×1,024ドット、表示画面：768×512ドット

●この画面は広告用に作成した、機能を説明するためのイメージ画面です。また、各種アイコン等は、SX-WINDOW ver.3.0がもつ機能を使って作成したもので、標準装備のものとは異なるものもあります。
●本広告中のエディタで表示している文字のフォントはZeit社の、「書体倶楽部」のフォントを使用しています。

シャープ株式会社　●お問い合わせは…コンシューマーセンター西日本相談室　〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部システム機器営業部　〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 CZソフト Oh!X 3体

に見たGUIの新展開。

❶マルチフォントエディタ編集例。文字ごとに文字種、文字の大きさの指定、修飾が可能で、イメージデータの貼り付けもOK。

❷CONFIG.SYSやAUTOEXEC.BATなどの編集に便利な「エディタ」モードの例。このように日本語マルチフォントエディタは、用途に合わせてカスタマイズできます。

❸❶の画面をプリンタで印字した例。対応プリンタも増えました。(カラー印刷は誤差分散により65,536色対応)

❹「パターンエディタ」で作成したデータを、背景に設定できます。

❺バージョンアップした日本語フロントプロセッサASK68K ver.3.0の辞書メンテナンスがウィンドウ上で可能。

❻アイコンデータや背景データを作成する「パターンエディタ」。文字の貼り付けなど、編集機能も一段とフレンドリーに。

❼オリジナルに作成したアイコンパターンの例。

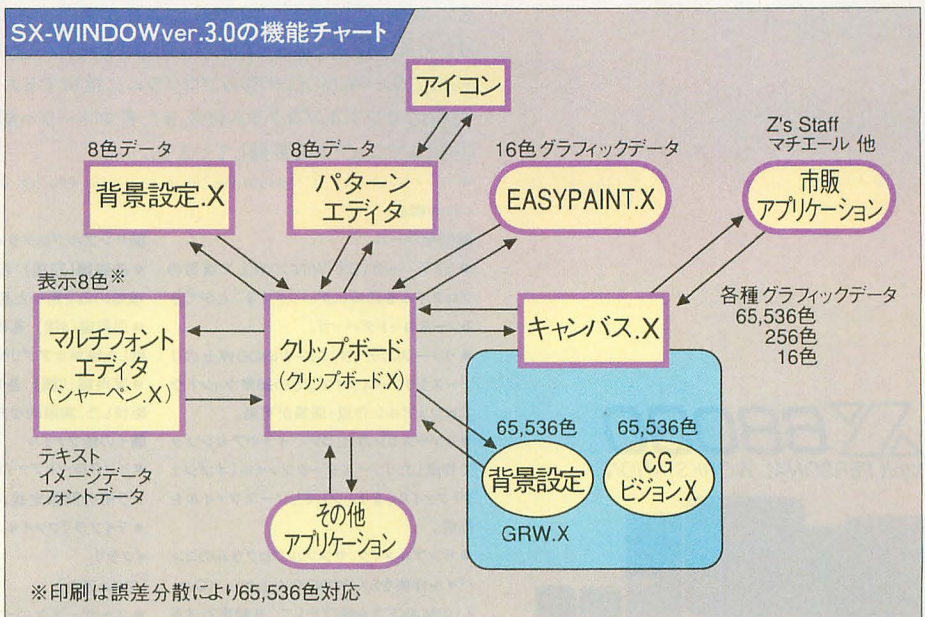
❽512×512ドットの範囲内で65,536色の表示が可能。

❾さまざまなグラフィックフォーマットに対応しています。

❿任意のサイズに縮小・拡大表示可能。

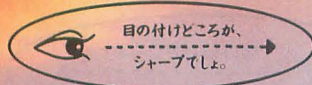
⓫異なる画像フォーマットへのコンバートができます。

⓬「CGAウィンドウ」、65,536色(最大)のコンピュータアニメーション表示が可能です。

発展性のあるプラットホームとしてのウィンドウシステム、
SX-WINDOW ver.3.0が提供する新たなGUI環境が
さらなるウィンドウ時代を予見する────。
国産オリジナルウィンドウとしての意味、未来への確かなビジョン、
ユーザーインターフェイスや高速化へのゆるぎない探求が
ここに凝縮されています。
65,536色表示はもちろん、さまざまな画像フォーマット対応、
イメージデータのコピー&ペースト、
動画、音楽/音声再生をサポートするマルチメディア環境。
そして、何よりもこれらが密接に連携して
統合的にハンドリングできるエキサイティングな環境を創造しています。
未来を照準に入れたウィンドウアーキテクチャ、
そのインテリジェンスがいよいよX68030/X68000シリーズで享受できます。

X68030

X68030 Compact

X68000 XVI

X68000 XVI Compact

For X68030/X68000series APPLICATION SOFTWARE

◎定評のあるGUI対応のウィンドウワープロ。

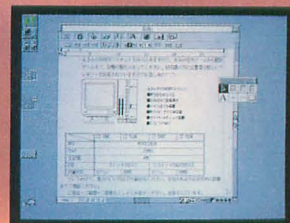
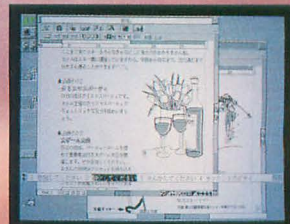
EGWord SX-68K

CZ-271BWD　2月発売予定

ウィンドウワープロとして評価の高いEGWordのSX-WINDOW対応版。キャラクタベースのワープロを超えたグラフィカル・ユーザーインターフェイス(GUI)による手軽なDTPソフトとしても優れた表現力を発揮します。定評ある日本語入力方式(EGConvert)によるインライン入力、文書互換を実現するEDF形式もサポートしています。

NEW

●禁則処理を生かした美しく、読みやすい文書作成：文字間隔を自然に美しく配置。さらに均等禁則など、豊富な禁則処理によるきめ細かな調整が可能。
●作図感覚で罫線・作表作業も快適：マウスによる作図感覚の作表、また豊富な線種で、行や列を気にせずに文字と文字の間に罫線が引けます。表組も自在に編集できるとともに、罫線で囲まれたブロック単位で網掛けも可能。
●DTPに迫る多段組、レイアウト表示：段組は2段から5段まで設定でき、段間隔の調整や段組線の表示はもちろん、自由な位置での改段も可能。レイアウト表示もOK。
●様々なグラフィックデータやテキストデータの貼り込みが可能：他のソフトで作成された色々な画像データ(GScript形式)をEGWordの文書に取り込めます。もちろん取り込んだ画像の編集もOK。
●短文・書式登録でルーチンワークの負担を軽減●充実した国語辞書機能に優れた専門辞書をプラス●実用性の高い逐次自動変換方式を採用●ウィンドウの特性を生かした優れたユーザーインターフェイス●ルーラによるスピーディ&イージーな書式設定●文書互換を実現するEDF(Extended Document Format)形式をサポート。

*5MB以上の空きのあるハードディスクが必要です。(4MB、ver.2.0)

◎待望のSX-WINDOW開発支援ツール。

SX-WINDOW 開発キット Workroom SX-68K

CZ-288LWD　2月発売予定

SX-WINDOW用のソフト開発に必要な開発ツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解ができる33種のサンプルプログラム付き。また各マネージャ解説と関数リファレンスの詳細なマニュアルも装備しています。

NEW

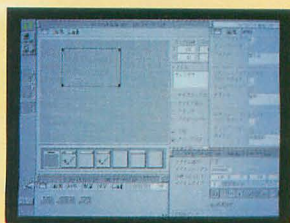
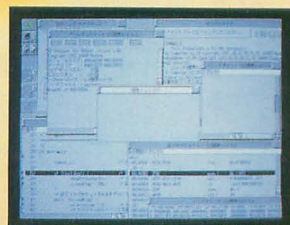
※メインメモリ4MB以上、SX-WINDOW ver.2.0以上、C compiler PRO-68K ver.2.1が必要です。

〈キット構成〉

■開発ツール
●SXデバッガ：SX-WINDOW上で複数のプログラムを同時にデバッグすることができるソースコードデバッガ。
●リソースエディタ：SX-WINDOW上のリソースをリソースタイプごとの編集ウィンドウでビジュアルに作成・編集が可能。
●リソースリンカ：Cコンパイラやアセンブラで作成したリソースデータファイル(オブジェクトファイル)をリンクしてリソースファイルを作成。
●サンプルメイク：サンプルプログラムのコンパイル作業をSX-WINDOW上から、XCver2.1のMAKE.Xを呼び出して、自動実行する簡易メイクユーティリティ。

■サンプルプログラム
●基礎編(23種)：各マネージャの基本的な機能のみを用いた基本動作の理解。
●応用編(4種)：基礎編での基本機能を応用した簡単なアプリケーションの作成。
●実用編(6種)：基礎/応用編での機能を駆使した、実用的なアプリケーションの作成。

■その他ファイル
●インクルードファイル：Cコンパイラとアセンブラ用の関数定義、データ定義ファイル。
●ライブラリファイル：Cコンパイラ用関数ライブラリ。

[マニュアル]
●ユーザーズマニュアル●プログラマーズマニュアル●SXライブラリマニュアル

その先のシーンへ。

●65,536色対応、動画ウィンドウ標準装備。

SX-WINDOW ver3.0 システムキット

CZ-294SS(130㎜FD)/CZ-294SSC(90㎜FD)各標準価格19,800円(税別)

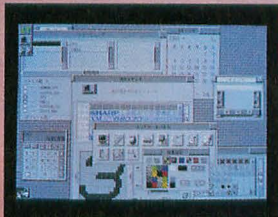
自然描画に迫る美しい表現が可能な65,536色表示のグラフィックウィンドウを装備。さらにグラフィックウィンドウ内でのアニメーション動画表示、各種グラフィックデータのコンバートも実現しました。またイメージデータの貼り付けなどをサポートした日本語マルチフォントエディタを始め、クリエイティブワークを支援する数々の便利機能を装備、Human68k ver.3.0システムディスクも付属しています。

※メインメモリ4MB以上必要です。

●「SX-WINDOW開発キット」のサポートツール。

開発キット用ツール集

CZ-289TWD　2月発売予定　NEW

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索するインサイドSX、イベントの発生を常時監視確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。（2MB、ver.2.0）

●SX-WINDOW対応ドローイングツール。

Easydraw SX-68K

CZ-264GWD　標準価格19,800円(税別)

ホビーからビジネスまで幅広い分野で活用できる、待望のドローイングツールです。イラスト、フローチャート、地図、見取り図など各種グラフィックが製図感覚で作成できます。作成したデータは、他のSX-WINDOW対応アプリケーションでも利用でき、企画書やプレゼンテーション資料の作成をサポートします。またレーザープリンタドライバを付属、このドライバはSX-WINDOW対応の他のアプリケーションでも利用することができます。（4MB、ver.3.0）

●SX-WINDOWを楽しく使うためのアクセサリ集。

SX-WINDOW デスクアクセサリ集

CZ-290TWD　標準価格14,800円(税別)

SX-WINDOWをさらに便利に、楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、アドレス帳、電子手帳通信ツール、パズルなど12種類の豊富なアクセサリが収められています。

1キーノート 2スクリーンセーバ 3スクラップブック 4ミュージックボックス 5ハイパーリンク(電子手帳通信ツール) 6アドレス 7スケジューラ 8ウィンドウアイコニファイ 9ソフトウェアキーボード 10パズル 11ファイルサーチ(ファイル検索ツール) 12フォントリンカ。（2MB、ver.3.0）

●マルチタスク機能をはじめ、通信環境がさらに充実。

Communication SX-68K

CZ-272CWD　標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションソフトを実行中でも簡単に通信が可能。また、ホスト局をクリックするだけの自動ログイン機能、初心者にも簡単なプログラム機能、最新モデム(20種類)もフルサポートしています。（2MB、ver.1.1）

●ウィンドウ対応グラフィックツール。

Easypaint SX-68K

CZ-263GWD　標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。（2MB、ver.1.1）

●FM音源サウンドエディタ。

SOUND SX-68K

CZ-275MWD　標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更できるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。まさにミキサー感覚で音創りが楽しめるツールです。（2MB、ver.1.1）

●SX-WINDOW対応になってさらにパワーアップ。

倉庫番リベンジ SX-68K ユーザー逆襲編

CZ-293AW(130㎜FD)/CZ-293AWC(90㎜FD)各標準価格6,800円(税別)

倉庫番10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。AI機能やエディット機能、キャラクタ変更機能も装備。半年で解けたらあなたは天才?です。（2MB、ver.1.1）

●X68030/X68000対応　NEW

NEW KIT

CZ-295LSD　標準価格44,800円(税別)

※メインメモリ2MB以上が必要です。

※C compiler PRO-68K/ver.2.0/ver.2.1をお持ちの方には有償グレードアップサービスを行います。

C compiler PRO-68KのX68030/X68000対応版。MPU68030、MC68882の命令セットに対応したアセンブラ、デバッガ、ソースコードデバッガを付属。またHuman68k ver.3.0、ASK68K ver.3.0にも対応。新たにGPIBライブラリ、MC68882対応フロートライブラリを付属しています。

※（2MB,ver.1.1）の表示は、メインメモリ2MB以上、SX-WINDOW ver.1.1以上が必要であることを示します。

※発売予定のソフトの画面は実物とは異なる場合があります。

●お問い合わせは…シャープ㈱電子機器事業本部(液映)システム機器推進プロジェクトチーム　〒162 東京都新宿区市谷八幡町8番地　☎(03)3260-1161(大代表)へ　シャープ株式会社

資料請求券

X68000 X68030
オリジナル3Dポリゴン
アクションシューティングゲーム

MIDI/内蔵音源対応
要2Mバイトメモリ
●Roland社 [CM-300 CM-500] [SC-55/mkII SC-155]

加速する戦慄は止められない。

ジオグラフシール

※画面写真は開発中のものです。

3D-RFS
With
AIR PERSPECTIVE
空間での自由な行動が可能

3.5インチ版タケルにて発売予定!!
※3.5インチ版のお問い合わせはタケル事務局へ

〈発売元〉

〒950 新潟市小張木219-10
☎(025)284-7304

お求めは、全国パソコン専門店で。取扱店がない場合は、住所・氏名・電話番号と商品名を明記し、価格に消費税を加算の上、当社宛に現金書留にてお申し込み下さい。
※5インチ版のみの販売になります。

卒業 Graduation

―グラデュエーション―

好評発売中!

あの女子高生育成シミュレーション「卒業」が、いよいよX68000に登場! 多感で青春まっさかりの女生徒5人。彼女達を無事卒業までみちびいていくのが、教師であるあなたの使命。5人の生徒にかこまれた1年間を、あなたはどうすごしますか・・・?

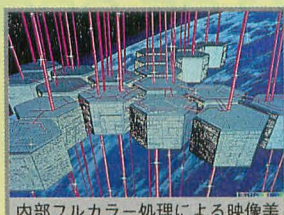
■PC-9801版からの完全移植
美しいグラフィック、彼女達のセリフ…。すべてをPC-98版から忠実に移植しました。しかも画面デザインはX68000の機能に合わせてアレンジ。SEIKAのロゴが、画面をオシャレに彩ります。

■MIDI対応になりました!
X68000版はMIDI(GS音源)対応。だからBGMの迫力、美しさが違います。臨場感あふれるサウンドが、いっそうゲームを引き立てます。

●X68000版は画面のデザインが変更になります。

TAKERU価格 ¥7,800(税込)
■対応機種:X68000/X68030 要2Mバイト
■企画・制作:TAKERUソフト

インダストリアルマジック
ハイパーピクセルワークス

多彩な機能が、バックアップ

内部フルカラー処理による映像美

ハイスピード・クイックレスポンスの快適なウィンドウシステム、アンチエリアスプリミティブレベル対応、内部フルカラー表現、高速・高性能ルーペシステム、多機能エフェクト、プリミティブ透明度処理、リアルタイムアンドゥ、リアルタイムターゲット、光学処理、プリミティブ合成処理、エクステンション機能、・・・CGは、新たな時代を迎えた。

TAKERU価格 ¥19,800(税込)
■X68000,030シリーズ/要2Mバイト
■企画・制作/ハイパー

オプション

「エクステンション#1マルチフォントシステム」
¥1,500　好評発売中!

R.Cロボット集+α vol.1

好評
発売中

ロボットコンストラクション「R.C.」のロボットデータ集です。第1回郵送ロボットバトル大会の結果報告と、その参加ロボットを中心に収録。オンラインで行われた「ロボットバトル大会」第1回、第2回で、ベスト8に残った強豪ロボットも収録しました。更に、「R.C.」がもっと使いやすくなる、ユーザー作の秀作フリーウェアも入ってこのお値段!「R.C.」ユーザーの方はぜひ手に入れてください。

TAKERU価格 ¥1,800(税込)
■対応機種　X68000,X68030/シリーズ 「R.C.」本体が必要です。
■企画・制作　エレクトリックシープ

ブラザー工業株式会社
〒467名古屋市瑞穂区苗代町2番1号
TAKERU事務局 (052)824-2493
東京営業所(03)5203-7133
大阪営業所(06)258-3024

通信販売 1994年4月1日より、送料/手数料が有料になります。
ソフト名、機種名、メデアのサイズ、住所、氏名、電話番号を明記の上TAKERU事務局まで現金書留でお申し込みください。送料/手数料は、1回のお申し込み総金額が5,000円以上の方は無料。4,900円までの方は500円をいただきます。4,900円までの方は現金500円をプラスしてお申し込みください。誠に勝手ながら、皆様のご理解とご協力の程、お願い申し上げます。(実施日'94年4月1日より)

ツクモパソコン本店II 3F、OAシステムプラザ（札幌・仙台・横浜西口・大須・大阪日本橋・岡山・広島・福岡・鹿児島）でも販売中／

宍戸留美

夢は、い・た・だ・き・よ……

コンパクト XVI 改造機。弊社にて1年保証。クロックは10／16／24の3モード。16／24MHzは背面トグルスイッチにより切替。RED ZONEの24MHz では正常動作しないソフト等がありますが、10／16MHz でご使用になれます。

X68000 Compact XVI 改

RED ZONE ¥160,000

・シャープ製CZ−6FD5完全コンパチブル・オートイジェクト機能付・ドライブ番号切替スイッチ付・木製（ナラ材）フロントパネル・対応機種／CZ−674C／300C／310C／500C／510C
・カラーリングオプションはプラス5,000円です。

シャープ製CZ-6FD5完全コンパチFDD(MK-FD1)

満開式軟盤駆動装置壱號

¥39,800(税別、カラーモデル¥44,800)

Compact XVI 改　5インチFDD
RED ZONE + MK−FD1
セット価格 **¥180,000**（税別）
（FDカラーモデルは＋¥5,000）

新登場 満開式硬盤駆動装置弐號
1GバイトSCSI ハードディスクユニット
MK-HD2
¥150,000→**¥125,000**（税別）
平均アクセスタイム8.3msの高速HDがこの価格
直販のみです。

98バスマウスを68で使っちゃうアダプタ
MOUSEJACK68-98
上記パソコンショップでもお求めになれます。
MK-MJ1 ¥4,000（税別）

当ショップは通販専門店です。X680×0用各種ハード・ソフトも取り扱っております。お電話にて商品リストと注文書をご請求ください。RED ZONEのご購入には承諾書が必要です。合わせてご請求ください。

〒171 東京都豊島区長崎1−28−23Muse西池袋2F
TEL (03)3554−7441　FAX (03)3554−3856

パソコンショップ満開
㈱満開製作所

X68030 Inside/Out

桒野雅彦◉著

●B5変形判・X68030回路図付き●予価2,600円

[本書の概要]

- ●X68000とX68030の違い
- ●CRTコントローラ
- ●数値演算プロセッサ
- ●エリアセット
- ●システムポート
- ●X68030の拡張スロット
- ●X68030の主要タイミングの実測結果
- ●仮想記憶とMMU

X68k Programming Series #3

X680x0 TEX

近刊

吉野智興/川本琢二/山崎岳志/実森仁志◉著

●B5変形判・2冊組・ビニール箱入り●5"FD7枚組●予価9,800円

待望久しいX680x0版TeXの解説書。

はじめてTeXに触れる人向けには簡単インストールプログラムを付けてあるほか、すでにTeXを使っている人向けには自分の環境にあわせたカスタマイズのしかたの説明もしています。

また、リファレンス編では

TeX、FONTMAN、PREVIEW.X、PRINT.X、MAKEFONT、METAFONTの環境変数・オプションの説明などもあります。

●ディスク 1

install. x……………インストーラ本体

texmain. 1zh…………TeXを動作させるために必要なファイル群

●ディスク 2

texbin. 1zh……………¥TeX本体やフォントマネージャ、プレビューワ等の実行ファイル

mf. 1zh………………METAFONTを作成する場合に必要なファイルのすべてを収録

●ディスク 3

pk118. 1zh……………画面表示用の118dpi PKフォント

pk400. 1zh……………プリンタ用の400dpi PKフォント

●ディスク 4

pk180. 1zh……………プリンタ用の180dpi PKフォント

pk360. 1zh……………プリンタ用の360dpi PKフォント

●ディスク 5

pk240. 1zh……………プリンタ用の240dpi PKフォント

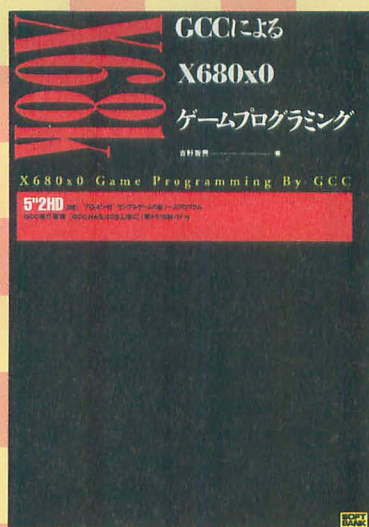
pk300. 1zh……………プリンタ用の300dpi PKフォント

●ディスク 6

freefont1. 1zh…………フリーの日本語フォント mincho46. bm1とmincho24.bm1

●ディスク 7

freefont2. 1zh…………フリーの日本語フォント gothic32. bm1とmincho32. bm1

＊なお、ディスクの内容は変更になる可能性があります。

SOFT BANK ソフトバンク株式会社出版事業部 〒103 東京都中央区日本橋浜町3-42-3 電話03(5642)8101

GCCによる
X680x0
ゲームプログラミング
X680x0 Game Programming By GCC
5"2HD

新製品紹介

OS-9/X680x0 ビデオPC for X680x0

Tan Akihiko 丹 明彦

X68000/030が本格的なマルチメディアの世界に足を踏み入れた。昨年誕生したばかりのデジタル動画規格「ビデオCD」対応ソフトを，他機種にさきがけてX68000/030で楽しむことができるようになったのだ。

概要

ビデオPC for X680x0は，X68000/030でデジタルビデオ(MPEG圧縮された映像・音声データ)の再生を行うためのハードウェアおよびソフトウェアである。そこそこのコストでフルモーション映像と音声を再生することができる。CDに長時間の映像を記録できる「ビデオCD」規格に対応しており，今後現れてくるであろう映画などのさまざまな映像ソフトへの展開が期待される。

ビデオPC for X680x0はOS-9/X680x0専用である。パッケージの内容は，

●ハードウェア

・MPEGビデオ/オーディオデコーダボード

●ソフトウェア

・MPEGデコーダボードのドライバ
・CDドライバ
・Player shell（パーソナルウィンドウ版）

●その他

・サンプルCD
・マニュアル

となっている。

ビデオPC for X680x0を利用するためにはOS-9/X68000 V2.4またはOS-9/X68030 V2.4.5が必要である。ビデオCDなどのCD-DVソフトを再生するためにはCD-ROMドライブも必要である。

対応ソフト

ビデオCDソフトは現在のところはほとんど存在しないが，これから動きが騒がしくなってくるだろう。

パラマウント社（米国）の映画CDがフィリップス社（オランダ）の製品として発売される。CD-I/FMVのタイトルとしては「トップガン」や「レッドオクトーバーを追え」などが制作されている。

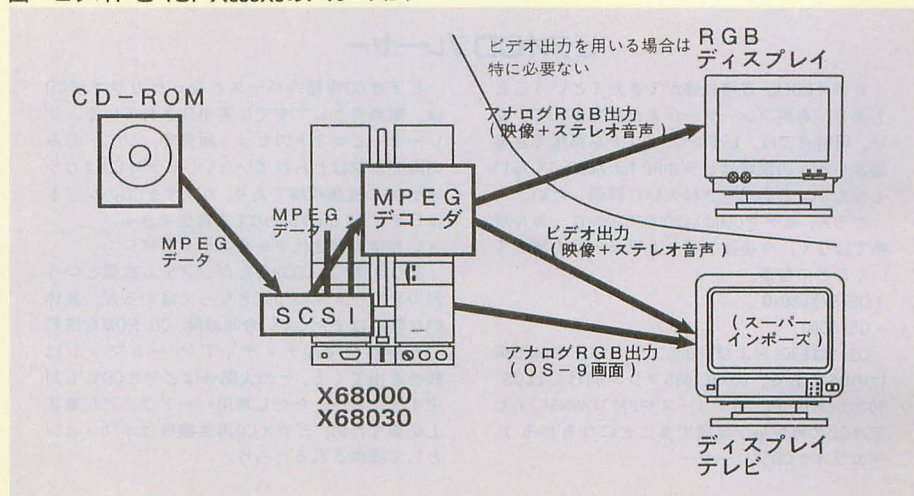
日本ビクターからはカラオケCDが発売される予定である。

ビデオPC for X680x0のパッケージにはAlias Japan社提供のサンプルCDが付属する。同社の3次元CGソフトを使って制作された映像をふんだんに詰め込んである。主に米国で制作されたTVコマーシャルフィルムだが，「ターミネーター2」「アビス」のSFXシーンなども入っている。

Video PC for X680x0のメカニズム

ビデオPC for X680x0は，CD-DVプレーヤーをコントロールするというものよりはもう少し柔軟なつくりになっている。

X68000/030から見れば，MPEGデータはCD-ROMにある通常のファイルとなんら変わらない。このMPEGデータをSCSIを通してX68000/030に読み込み，I/Oポートを介してMPEGデコーダボードに流し込む。

図 ビデオPC for X680x0のメカニズム

デコーダボードに送るのはMPEGデータであれば特にメディアを選ばない。CD-ROMでもハードディスクでもMOでもよい。ただ，MPEGデータはそれなりに大きなデータであるし，現状ではエンコーディングの環境がないこともあり，ソースとしてCD-ROMが事実上標準である。現状流通しているMPEGデータのソースの大半はCD-ROMに格納されているはず。

SCSIの転送速度の関係上，問題なく使用できるのはXVI以上ということになるかもしれない。

MPEGデコーダボードにMPEGデータを転送すると，ボードから映像信号と音声信号が出力される。映像はアナログRGBまたはNTSCコンポジット，音声はステレオ。ボード後端には15ピンのアナログRGBコネクタとRCAコネクタ×3（映像，音声×2）が配置されている。

このメカニズムからわかるように，実際に出力される映像と音声はX68000/030本体に取り込まれることはない。

対応CD-ROMドライブ

世の中には多くの種類のCD-ROMドライブがあるが，ビデオPC for X680x0で使えるのはそのうちの一部である。

・ドライブの速度

フルモーションビデオは，当然のことながらリアルタイムで再生を行わなくてはならない。転送速度（ビット転送レート）を確保するために，倍速以上のCD-ROMドライブが必要ということになっている。

・ドライブの対応フォーマット

CD-ROMドライブによってはビデオCDを受け付けないことがあるということは，注意しておく必要がある。

CD-ROMドライブのなかには，メディア挿入時にそのタイプをチェックして，対応していないフォーマットのCDをイジェクトする機能を備えているものがある。これはファームウェアのレベルで実現されているので，ソフトでは回避できない。

現時点で，東芝のCD-ROMドライブはCD-Iを受け付けない。計測技研から発売されているCD-ROMドライブがこれにあたる。普及という観点からみた場合，これは少々頭の痛い問題になるかもしれない。

ソニーとパナソニックのCD-ROMドライブでは動作が確認されている。ロジテックのCD-ROMドライブも中身はソニーの製品なので問題ない。

個人的にはパイオニアのLaserMemory，つまり6連装CD-ROMにとても興味を持っている。映画のタイトルが出てくると，どうしても2枚連続再生は欲しいし，カラオケCDを利用する人ともなると6連装ですら足りはしないだろうから。

使用感

MPEGデコーダボードはX68000/030のI/Oスロットに装着する。基板が本体から1cmほどはみ出すが，気にはならないだろう。

ビデオPC for X680x0のパッケージには，OS-9のパーソナルウィンドウ上で動作する再生プログラムが付属している。再生，ポーズ，コマ送り，スキップサーチなどの操作をマウスで行うことができる。

X68000/030の純正ディスプレイテレビであれば，再生画像をビデオ端子に，コンピュータ画面をアナログRGBに入力して，再生時にはスーパーインポーズした状態でポーズや早送りなどの操作を行うという使用形態が一般的になるだろう。

気になる画質であるが，NTSCと思えばまずまずの画質。MPEG特有の（というかDCT系の画像圧縮に共通した）ノイズが生じはするが，許せないほどのレベルではない（もちろん個人差もある）。ビデオ出力で十分。アナログRGB出力を利用しても，そう極端に画質が上がるわけではない。

なお，映像出力はフルカラーであり，X68000/030から制御するといっても65536色に制限されるということはない。X68000/030が扱うのはデコード前のMPEGデータだけであり，出力は扱わない。だから当然の話ではある。

デジタルオーディオをステレオ再生できるというのも強調しておこう。MPEGの圧縮フォーマットは音声圧縮も規定している。ミニディスクよりもう少し圧縮比が高いということである。

ビデオCDの可能性

ビデオCDは映像と音声を1枚のCDに最大74分収録できる。そもそもCDが74分の音声を記録するためのメディアだったことを思えば，まさに驚異の技術である。

画質的にはLDが上だろう。エンコーダの違いでかなりばらつきがあるが，なんといってもメディアの小ささが魅力。たいていの映画が2枚組のCD-ROMに入ってしまうというのもすごい。LDは片面60分，両面で120分記録でき（CLVの場合。CAVはその半分），メディア1枚あたりの再生時間はLDが長いのだが，CD-ROMは片面なので，そういう比較はあまり意味がない。むしろ，映画には「2時間ちょっと」というのがやたら多いので，片面60分と74分の差は意外に重要だったりするのである。たとえば2時間20分の映画はLDでもビデオCDでも2枚組。メディアの価格や携帯性からいってもビデオCDはかなりいいセンいっていると思う。あとは，ソフトがどの程度出てくるかという1点にかかっている。

CDというメディアは，プログラムもデータも音楽も映像も入るわけで，なかなかにおいしいメディアだと思うのである。製造工程が同じというのもうれしい。

ビデオPC for X680x0の意義

再生に専用のハードウェアが必要とはいえ，ビデオCDのパフォーマンスは一般のAV機器と肩を並べるものであり，たとえ

ビデオCDプレーヤー

ビデオCDは，規格自体ができたてということもあり，専用プレーヤーがまだ発売されていない。現時点では，ビデオCDソフトを再生できる製品レベルの環境はビデオPC for X680x0くらいしかない。この迅速さは大いに評価したい。

むろん，ビデオCDはX68000/030のローカル規格ではなく，今後はプラットフォームも増えていくことになる。

・OS-9/X680x0

・OS-9000

OS-9は6809および68000ファミリーのMPU向けのOSである。80386/486マシン向けにはOS-9000があり，PC-98シリーズやFM TOWNSにもビデオCDの再生系が登場することになるだろう。

・カラオケCDプレーヤー

ビデオCD規格のベースとなったカラオケCDは，業務用としてすでに実用化されている。プレーヤーとソフトのセット販売で，ソフトのみの販売形態はとられていない。ビデオCDはカラオケCDの拡張規格であり，カラオケCDのソフトはビデオPC for X680x0でも再生できる。

・いわゆる新世代マルチメディアマシン

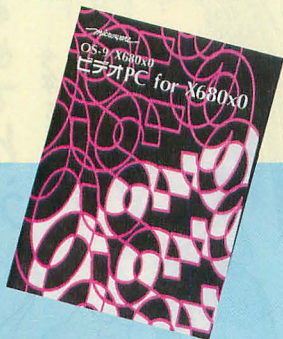
3DOやAMIGA CD32などがシステム拡張という形でサポートする予定となってはいるが，具体的な製品はまだない。今年以降，CD-ROMを搭載した新世代マルチメディア/ゲームマシンは続々と出てくる。その大部分はビデオCDにも対応するだろう。ただし専用ハードウェアが事実上必要なため，ビデオCD再生機能はオプションとして提供されるだろう。

デコーダボードを装着したところ

移動可能なメニューバーが表示され，再生，ポーズ，コマ送り，スキップサーチなどの操作はマウスで行う。これはコマ送り

ばQuickTimeのような「よくぞコンピュータでここまで」とただし書きをつけられて紹介されるようなものとは一線を画する。むろん，両者はまったく異なるものであり，こういう比較は無意味かつ卑怯であるが，エンドユーザーになりきった場合，やっぱりQuickTimeの画面は小さくて物足りないか，さもなくばコマ落としのぱたぱたアニメーションでしかないのだ。なおQuickTimeの名誉のために申し上げておくが，QuickTimeでもフルスクリーン・フルモーションのアニメーションは可能である。ただしそれには，Quadra級のマシンと巨大なRAMを必要とする。

＊　＊　＊

マルチメディアの波になんとなく乗り遅れ，なかば開き直ったかのごとくCD-ROMの世界を拒絶しているようにもみえるX68000/030の世界だが，ここにきて一気にリードしたとさえいえる。ビデオPCが，他機種の後追いでないマルチメディアのシステムをX68000/030に真っ先に供給したという点はここで強調しておきたい。

X680x0用　58,000円（税別）
マイクロウェア・システムズ
☎03(3257)9000

著作権

デジタルオーディオ/ビデオ特有の問題として，著作権の問題は常につきまとう。

デジタルメディアがマルチメディアともてはやされ，誰もがプロの扱う素材と同等のものを品質の劣化なしに入手できる時代がやってくると，違法コピーし放題ということになってしまいかねない。

そこで，デジタルのままでは無限にコピーできないように技術的または政治的な対抗手段がいくつかとられてきた。たとえばDATのデジタルコピーは1世代までしかできなかったし，ごく最近までオーディオCDの音声データをCD-ROMドライブを介して読み出すことは不可能であった。

ビデオCDについても事情は同様で，MPEGデータそのものはコンピュータからはバイナリデータとして見えるけれども，それをデコードしたあとの映像を計算機のメモリに取り込むことは禁じられている。

もっとも，MPEGデータの場合，その気になればファイルをデコーダに送らずにソフトで直接デコードすることもできる（処理は非常に重いためリアルタイム再生は難しい）ので，まあ紳士協定みたいなものではある。

デジタル・ビデオ関連用語

・MPEG

Moving Picture Experts Groupの略で，デジタル動画像の標準化を行うワーキンググループ。

・MPEG1/MPEG2

同グループが定めた動画像圧縮フォーマット。その考え方はかなり強烈である。個々のフレームを圧縮するだけでなく，フレーム間の情報も利用する。画像の中で動いている物体を検出することによって，フレーム間の差分情報を節約するという。これにより，1枚のCDに数10分の映像を記録するという離れ業を達成している。

MPEG1は，主にコンシューマを対象としたもので，画質はそこそこだがデータ量が少なくてすむ。ビデオPC for X680x0はMPEG1フォーマットで圧縮された映像/音声データを再生できる。

MPEG2はまったく異なるフォーマットで，放送を想定した規格である。ハイビジョンでの使用にも耐え得る画質を持っているという。ただしデータ量もそれなりで，サンプリングレートが高く，CD-ROMにも5～10分しか記録できないという(10倍速級のCD-ROMドライブが必要になるのだろう)。年内には規格が固まるという。

・エンコード(encode)/デコード(decode)

日本語で符号化/復号化と訳される。MPEGの場合は圧縮/伸長といってもよい。生の映像データをCDやハードディスクに入るようにMPEGデータに変換することをエンコード，MPEGデータを再生時に元の映像データとして復元することをデコードと呼ぶ。ビデオPC for X680x0はこのうちデコードだけを行う再生専用システムである。なおマイクロウェア・システムズでは，X68000/030向けのエンコードシステムも販売する予定。さらに，映像ソースからビデオCDを委託作成する業務も行う予定とのこと。

・ビット転送レート

MPEG1の再生画像の画質はそれほどいいとはいえないが，それは限られたビット転送レートのなかで動画像の情報を詰め込まなくてはならないためである。

映像は当然のことながらリアルタイムで再生しなくてはならない。一方，CDから1秒間に転送できるデータ量には限りがある。この差が圧縮率を，ひいては画質を規定する。ビット転送レートをちょっと上げるだけで画質は格段に向上する。それは素人目にもわかるほどの違いがあるという。むろん，こうした高画質のビデオCDの実現のためには，2倍速CD-ROMドライブでも不足かもしれないし，1枚のCDに74分の映像を格納することも不可能になるだろう。

なお，限られたビット転送レートのなかでも，エンコーダのアルゴリズム次第で，ある程度画質を上げることができる。画質は主に動き検出のアルゴリズムを洗練することで向上する。動きの激しい映像ではエンコード後のデータ量が大きくなりがちなのだが，動き検出がうまくいけばデータ量を抑えられるのである。なお，動き検出のアルゴリズムが改良されても，MPEG1としての記録フォーマットは変わらないので，プレーヤーを交換したりする必要はない。

・DCT

離散コサイン変換(Discrete Cosine Transform)。フーリエ変換などと同じような直交関数系を用いた画像圧縮に多用される演算アルゴリズム。SX-WINDOWのキャンバス.Xでもサポートされている JPEGフォーマットはDCTを用いたメジャーな圧縮フォーマットのひとつ。

・CD-DV

Compact Disk Digital Videoの略。CDに記録されたデジタルビデオ。

・カラオケCD

フィリップス社(オランダ)と日本ビクターによる映像圧縮の規格。業務用として実用化されている。「枯れた規格」といって差し支えない。

・ビデオCD

カラオケCDをベースにして，静止画再生機能(動画像のものより高画質の静止画を記録/再生する)とプレイバックコントロール機能(メニュー再生機能，映像の再生順序をプログラムできる)をオプションとして追加した規格。松下電器産業，ソニー，日本ビクター，フィリップスによる。

・CD-I

Compact Disk Interactive。フィリップスとソニーの提唱による対話型メディアの規格。音楽と映像と文字を12cmCDに詰め込んでいる。CD-Iは専用のプレーヤーで再生するものである。ビデオPC for X680x0ではCD-Iそのものは再生できない。

・CD-I/FMV

CD-IのFull Motion Video拡張規格。ビデオPC for X680x0はCD-I/FMVに対応したソフトを再生できる。

CD-DVのロゴマーク

OS-9/X680x0ビデオPC for X680x0

illustration : Y.Kawahara

▲横井　賢一（富山県）

▲森井　浩之（兵庫県）

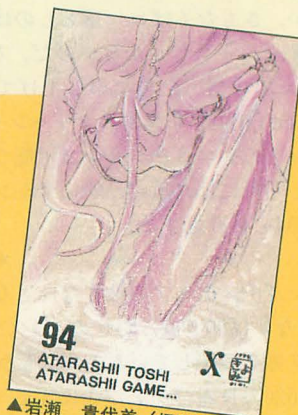
▲八尾　唯仁（神奈川県）

Oh!X reader'sぎゃらりぃ

今年も大盛況のOh!Xreader'sぎゃらりぃ（年賀状編）
去年に引き続き，今年もOh!Xをよろしくお願いしますね
次回のreader'sぎゃらりぃは5月号。力作を期待しています

▲藤原　彰人（岡山県）

▲帝釈　好孝（三重県）

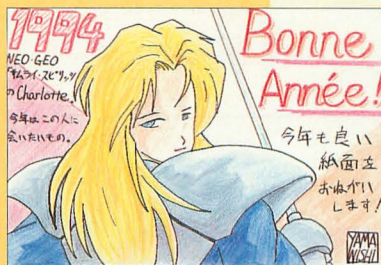
▲占部　哲彦（広島県）

▲岩瀬　貴代美（福岡県）

▶武田　正道（兵庫県）

▲江端　一・栄子・加奈子（東京都）

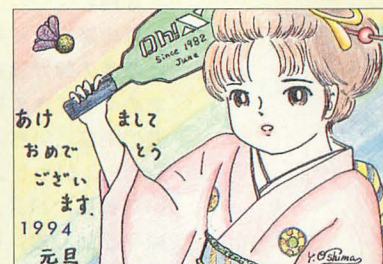
▲鈴木　貴久（神奈川県）

▲近藤　隆生（埼玉県）

▲青木　一師（奈良県）

▲山西　孝到（大阪府）

▶岡田　徹（神奈川県）

▲加藤　政宣（宮城県）

▲藤沢　実（東京都）

▲大嶋　靖浩（栃木県）

◀武田　和凡（京都府）

▲伊藤　健文（神奈川県）

▲藤田　英子（大阪府）

▲前田　基行（兵庫県）

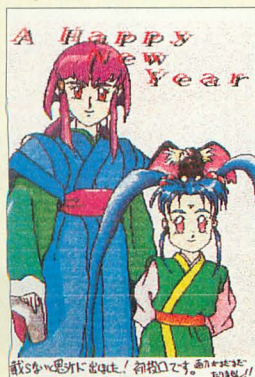
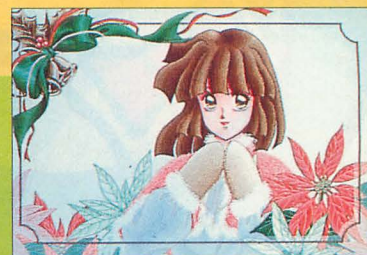
▲吉岡　洋明（埼玉県）

◀日比生　雅治（大阪府）

◀玉野　健一（奈良県）

◀▲瀬川　直子（千葉県）

［第34回］
イメージの種

江口 響子

響子 in CG わ〜るど

3月。

霜の降りなくなった黒い土の中から，ふっくりと緑色の芽が頭をもたげてくる，そんな季節がまためぐってきました。いま住んでいるところには，道の両脇にいちょう並木が続いていて，この頃になると，いっせいに芽吹いてきます。茶褐色の乾いた枝に，パステルで描いたような黄緑の色が点々と加わり，日ごとにその割合が増えてくる……やはり，春はいいですね。

そうはいっても，今年の春は，戦後最大の不況の風が吹き荒れていて，とても寒い。収入の不安定な絵描きという自由業の，私自身の生活も決して楽ではありません。でも，先行きのことを心配しすぎて，気持ちまでもが硬く冷たい冬の土になってしまうのだけは，避けたいのです。

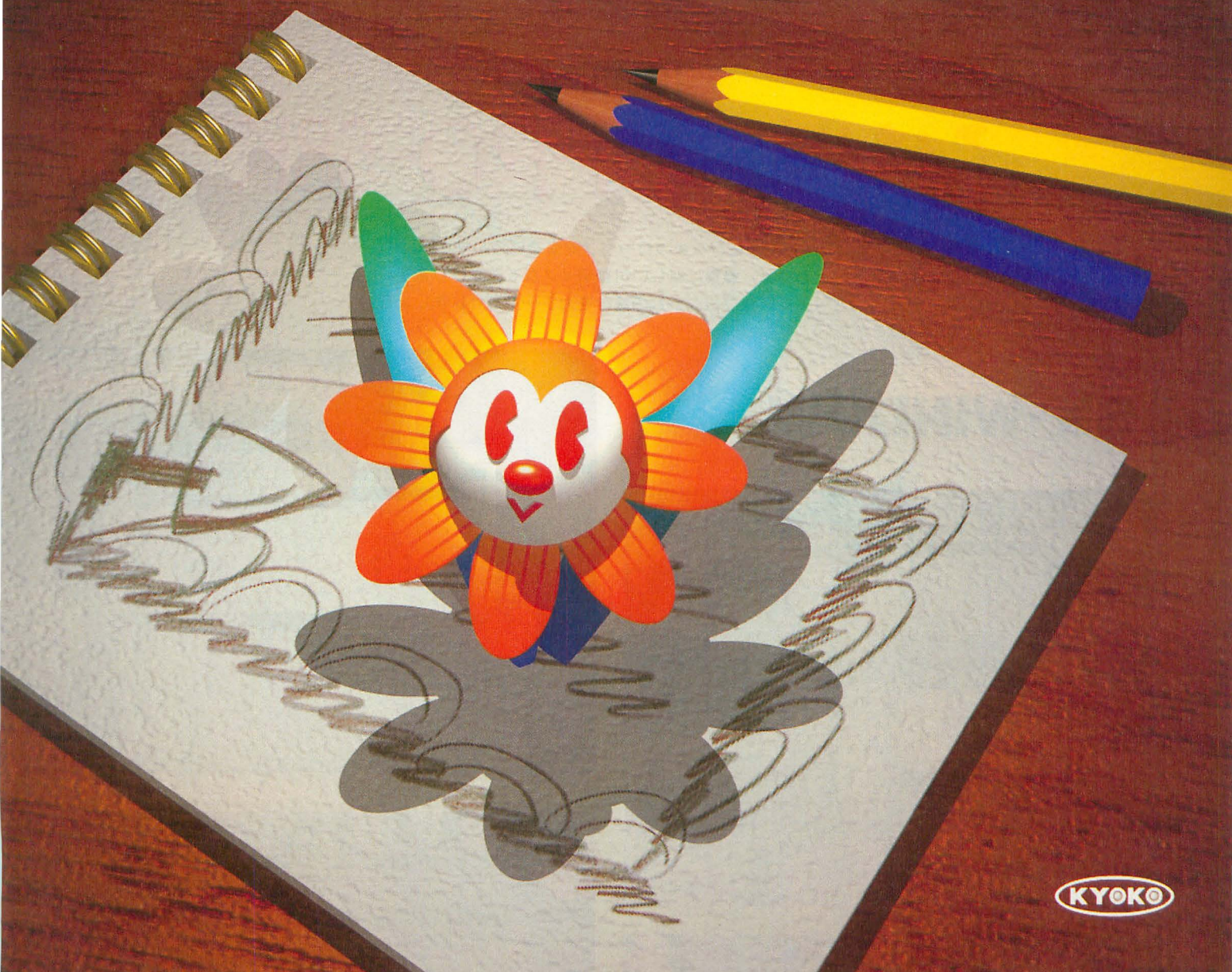
さて，ここにいくつかの種があります。どんな形をしているのか，何色なのか，目で実際に見ることはできません。が，イメージの種といって，誰でも持っているものです。それを，心のなかのいちばん柔らかい場所に，そっとまきます。土が踏み固められないように守りながら，想像力と技術力という肥料をやって育てていくと，思ってもみない大きな美しい花が咲くことがあります。ああ，こういう花を咲かせる種だったのかと，そのとき初めてわかるのです。

大切にするといっても，ときどきは冷たい世間の風にさらしてやります。ビニールハウスで囲ったままだと，最後にハウスから出したとたんにしおれてしまう可能性が高いのです。そして，気長にゆっくりとゆっくりと育てていきます。注意しないと，ようやく芽が出たと安心したとたんに，腐って，深い心の闇のなかへ沈んで見えなくなることもあります。

私は，キャラクターを考え，育てるのが好きです。もちろん，こうして言葉を使って文章を書くことも好きですが，それ以上にキャラクター・デザインに興味があります。

その種は，人間の気持ちです。嬉しかったり，悲しかったり，愛したり，妬んだりと，あらゆる人の感情を色や形にしてみたいと思っています。

何枚も何枚もスケッチを繰り返し，自分の感覚にぴったりと合う線を選びだします。その線が手に馴染んで描けるようになるまで，練習をし，完成にこぎつけるのです。

CGで描くときは，手描きの作業はスケッチにとどめておいて，あとは画面上でデータの修正を

繰り返します。コンピュータは黙々と描いてくれますから，時間の許すかぎり，数値の入れ換えを試みます。ただ，そのときはいつもこういうものを作りたいのだ，というはっきりとしたイメージを思い浮かべています。

出来上がったものは，自分では満足していても，ひとりよがりになってしまう可能性もあるので，人に見てもらって感想を聞きます。ただ，あまり人の顔色ばかりうかがっていると，オリジナリティがなくなってしまうのも事実。そのあたりは，バランスをとるのが難しいところです。

誰もが持っているイメージの種。皆さんはどんな種を持っているのでしょうか？

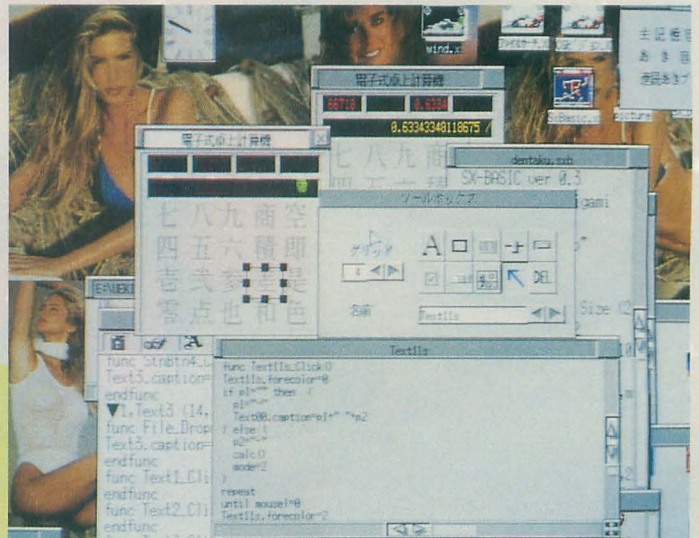
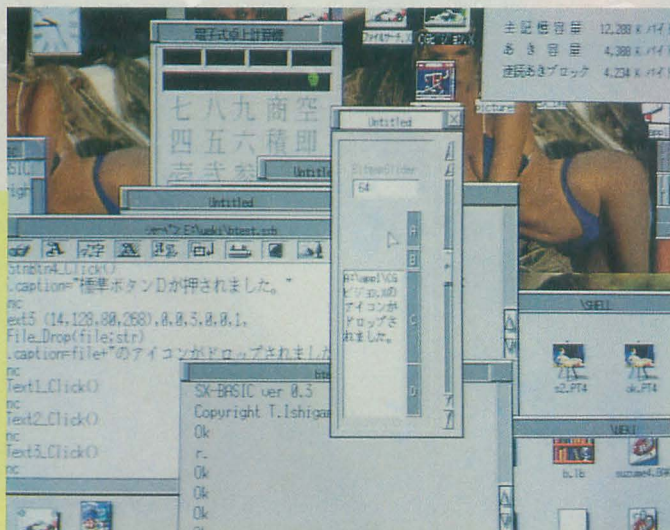
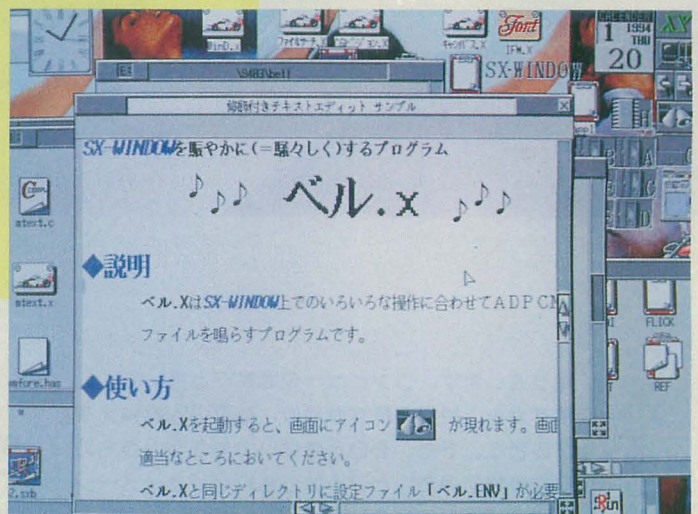
今回のCGデータ

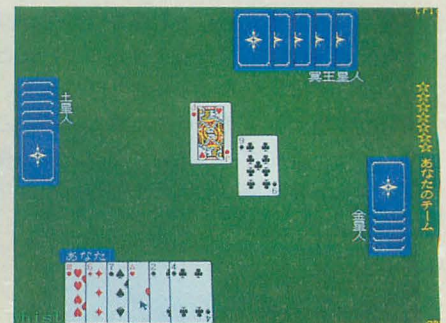
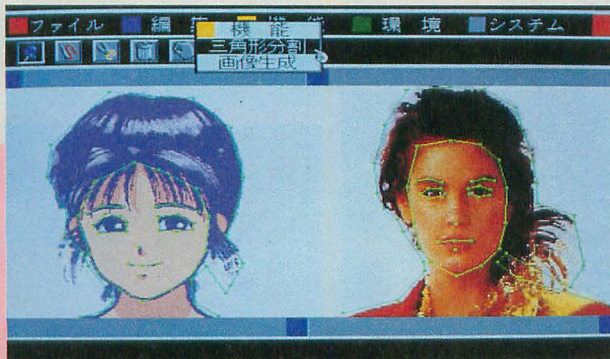
1280×1024ピクセル
1670万色フルカラーを4×5ポジで出力
総物体数　164　すべて，2次元曲面を集合演算
点光源　1　平行光線　1
使用ソフトは，C-TRACE
マッピングデータ作成には，Z's STAFF PRO-68KとMATIER
机の木目は，スキャナで取り込んだ512×512ピクセルの画像をテクスチャおよびバンプマッピングとして併用
スケッチブックの紙質はバンプマッピングで表現

【特別企画】

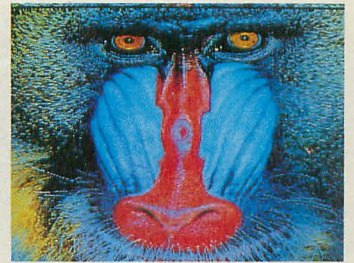
ひなまつりPRO-68K

3月だからひなまつり……（安直でごめんなさい）。
ともあれ，無事に新しい付録ディスクをお届けします。
さっそく収録プログラムの数々をカラーで紹介しましょう。

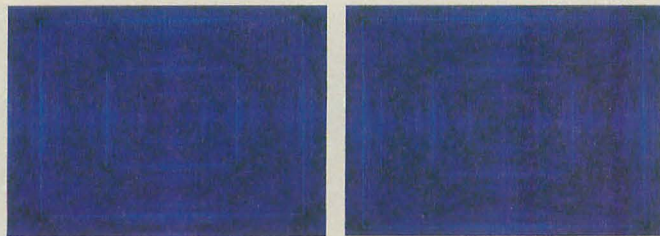
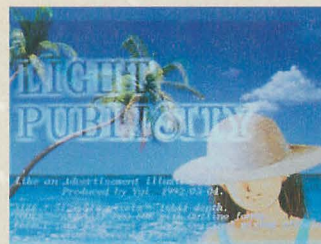
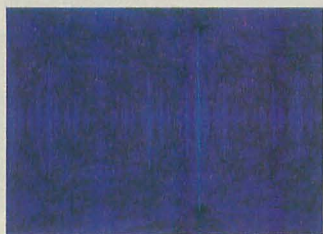
SX-WINDOW

SX-WINDOWで手軽なプログラミング環境を実現するのが，このSX-BASICです（背景は気にしないように）。ウィンドウデザイナで基本的な画面を設計し，SX-BASICインタプリタで実行します。画面入出力はウィンドウエンジンによって行われます。右はシャーペン仕様の拡張テキストを表示するMTEXT.Xです。

CARDGAME

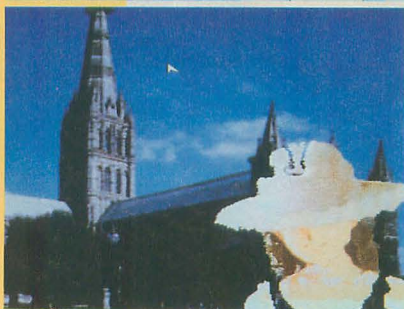
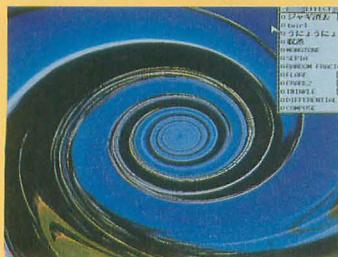
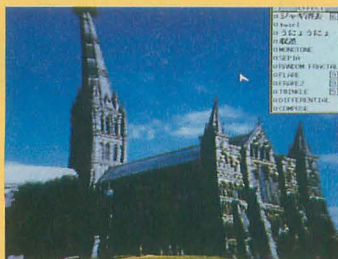
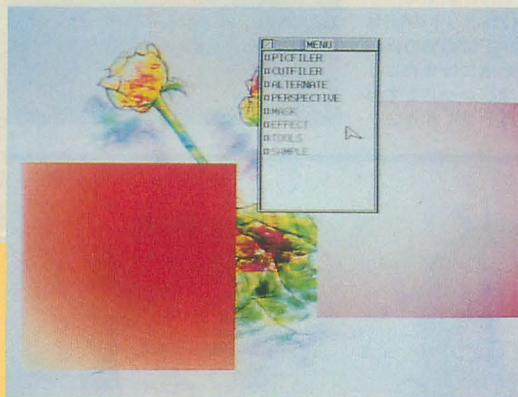
カードゲーム開発支援用のX-BASIC用関数ライブラリCSFとCARDSP.FNCのサンプルプログラムです。さすがにスプライト。カードが画面をびゅんびゅん飛んでいきます。サンプルはOldMaidとWHIST。OldMaidって……要するにババ抜きですね。

Morph!

２枚のグラフィックを滑らかにつないでみせるのがモーフィングです。対応する点をつなぎ三角形分割を実行してレンダリングします。画面横に出ているゴミは高画質モードではほとんど消えるので安心を。

AMISYSTEM

AMIはSCSIドライブを利用したアニメーションシステムです。ここではサンプル画像生成プログラムによって画面に水滴が落ちるさまをシミュレートしてみました。画面モード256×256ドットの65536色モードでの実行例。画像生成にはそれなりに時間がかかりますので注意してください。

アニメ顔の女の子から取り込み画像へのモーフィング実行例です。一般に途中パターン部分は崩れがちなのですが，かなり自然な仕上がりになっていますね。だんだんと女の子の年齢が上がっていくさまを見ている感じでしょうか。ちなみに画像出力は低画質モードで行い，画面端に出るゴミは修正されています。構図などのよく似たデータを使うことがコツといえるでしょう。

AMISYSTEM

AMIシステムのプログラムによるサンプル生成その2です。青1色のため写真ではちょっとわかりにくいのですが，画面には四角形の組み合わせによる再帰図形が魔法の回廊のように伸びており，そのなかをぐるんぐるんと動きまわります。データ作成も簡単なので，まずこのサンプルデータを動かしてみるのがよいでしょう。

波形その2。光源位置などを設定してさらに美しい波紋を描画します。波紋の反射などにも対応しています。当然，作成にかかる時間はいちばん長くなりますけど。

Z's-EX & MATIER-EX

Z's-EXとMATIER-EXの画面です。
上はMAGIC WANDの動作のようすを表したものです。ぼけぼけでエフェクトのかかりまくった画像に対してMAGIC WANDを実行したところですが，ちゃんと色の輪郭を取ってきていることがわかります。輪郭はマスクで示されるので，マスクペイントすれば領域を切り出すことができます。あとは合成など思いのまま。そのほか，ユーザーが作ったさまざまな処理も簡単に組み込むことができます。
左はもともとX-BASICで書かれていたものをCになおして組み込んだものです。特殊効果としかいいようがありませんが……。

SOFTWARE INFORMATION

そろそろ春の匂いが近づいてきました。サクラサクを待っている人はあとひとがんばりかな。闘いの季節はまだまだ続くようで，格闘ゲームも新作登場。人気ゲームの続編もいろいろ開発中とのことで期待大です。

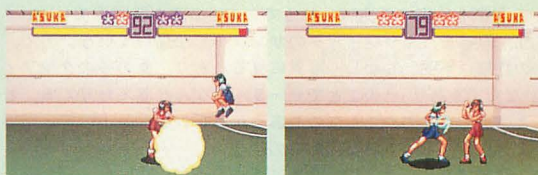
あすか120%

１月に久々の新作「マッドストーカーX68」を発売したファミリーソフトだが，早くも次回作の発売が決定した。

ジャンルは同じく格闘モノ。しかし，前作とはがらりと雰囲気を変えて，舞台は「私立繚乱女学院」。文武両道をモットーとし，トップレディの育成を目指す名門お嬢様学校である。そこで行われるのが，高等部名物「部対抗予算争奪戦メガファイトトーナメント」。そう，今度は闘うのは女子高生たちなのだ。

主人公は化学部の最終兵器「本田飛鳥」。１年生ながらも文化部でただひとり予選を突破した彼女は，悲願の優勝への期待を一身に背負い，ほかの５人との闘いに挑む。

開発は「マッドストーカーX68」と同じくフィルインカフェ。３月発売予定。

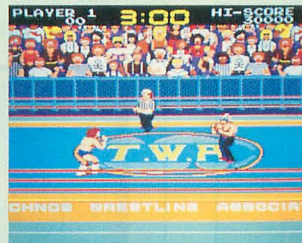
X68000用　5"2HD版　価格未定
ファミリーソフト　☎03(3924)5435

画面はFM TOWNS版の開発中のものです

エキサイティングアワー/出世大相撲

ビデオゲームアンソロジーシリーズ第８弾も格闘ゲーム。テクノスジャパンのアーケードゲーム「エキサイティングアワー」と「出世大相撲」を完全移植して１枚のディスクに収録，２月25日に発売される。

「エキサイティングアワー」は，1985年に登場したプロレスゲーム。レバーとボタンの組み合わせによる技を駆使して闘っていく。目標は輝かしい王者の証のチャンピオンベルトの獲得。

もうひとつの「出世大相撲」の内容は，まさにタイトルそのまま。相撲界の頂点を目指して勝負を重ねていく。幕下に始まり，レバーとボタンで横綱の地位を勝ち取るのだ。

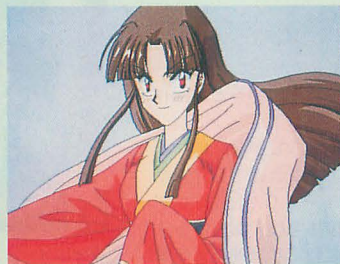
X68000用　5"2HD版　5,300円(税別)
電波新聞社　☎03(3445)6111

レッスルエンジェルス2

昨年８月に発売されたカード型対戦ゲーム「レッスルエンジェルス」の続編もただいま移植中である。

女子プロレスの華麗な世界での過酷な闘い。敗者の屈辱はお約束の脱衣シーン。勝者の栄光はチャンピオンベルト。

今回は自動モードが加わり，カード選びもコンピュータにまかせたままで観戦に徹することができるので，プロレス音痴のキミでももう大丈夫。プロレスにはちょっとうるさいぞ，という人のためには，技の外し方にバリエーションが加えられるなど，そっちもちゃんとパワーアップで安心だ。

前作と同じくグレイトによる開発。発売予定は３月だ。

X68000用　3.5/5"2HD版　価格未定
ブラザー工業(TAKERU)　☎052(824)2493

麻雀航海記

ブラザー工業(TAKERU)の次の新作ゲームは,「麻雀航海記」。シミュレーションゲーム「THE ATLAS」のパロディの麻雀ゲームだ。

「母国を代表する海賊となって,7つの海を征服する」ということだが,現在まだグラフィックのサンプルしか届いていないので,ゲームの詳細は不明。しかし,写真に見るとおり,女の子のグラフィックは「なんでもあり」とのことでセーラー服なんか着てたりする。ということは,あんなのやこんなのもあるかもしれない……ので,期待していよう。舞台もいろいろ変わるのかな。

開発はアレックスで,発売は4月の予定。

X68000用　3.5/5″2HD版　5,800円(税込)
ブラザー工業(TAKERU)　☎052(824)2493

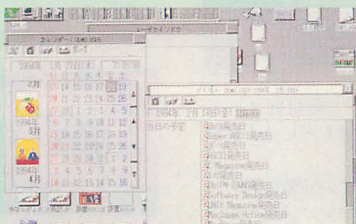
DoubleBookin'

計測技研より,SX-WINDOW用のスケジュール管理ソフトが発売された。

この「DoubleBookin'」は,短期・長期のスケジュールを記録したり,検索したりするだけではなく,SX-WINDOWのマルチタスク環境を活かして,設定時間にさまざまなイベントを起こすことができる。音楽演奏の開始や停止,テレビ画面への切り替え,シャーペンなどのソフトの起動など,ユーザーの必要と工夫次第で使い方はいろいろ。X68000そしてSX-WINDOWならではの快適環境を作れるだろう。

また,シャープの電子手帳やいま話題のPI-3000「ザウルス」とデータのやりとりをしたり,システム手帳用のリフィルをプリンタ出力するなど,携帯ツールとのリンクも図られている。

詳しい紹介は,来月号で行う予定。

X68000用　5″2HD版　12,800円(税別)
計測技研　☎0286(22)9811

デジタルアートコレクション vol.1〜8

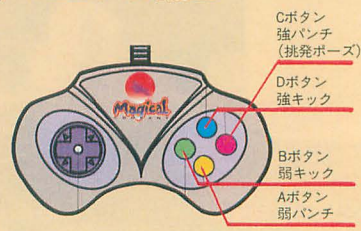
パソコン通信で発表されているアマチュアのCG作品を集めたデータ集が発売中。Oh!Xでも活躍中の川原由唯氏などのアマチュアCG界の人気作家の作品が多数収録されている。通信と異なり,ダウンロード時間の短縮などの配慮が不要なため,ネットでの配布のものに加筆,修正を行った作品もある。

現在,vol.1〜8がすでに発売ずみだが,今後も順次,続編が予定されている。奇数号は4096色中16色の作品集でPC-9801, FM TOWNS版もある。偶数号はX68000専用の65536色の作品集で,こちらはTAKERU版のみ。

パッケージ,TAKERU版に加えて,通信販売も開始された。

X68000用　3.5/5″2HD版　各1,500円(税込)
(TAKERU版)　各1,200円(税込)
vol.8(3枚組)は1,400円(税込)
CONNECTLINE　☎0899(26)7821
ブラザー工業(TAKERU)　☎052(824)2493

「餓狼伝説2」マニュアルの訂正

昨年12月に魔法株式会社より発売された「餓狼伝説2」ですが,マニュアル6ページの本体付属ジョイパッドの説明文において,パンチボタンとキックボタンの表記が逆転していました。正しくは図のとおりです。

宝魔ハンターライム7
宝魔ハンターライム8

最初は6話完結の予定だった「宝魔ハンターライム」シリーズだが，アニメファン好みのグラフィックに脳天気ストーリー，お手頃価格が好評につき，12話まで延長されている。第1話で登場したボスと第6話での対決だったが，第7話以降ちょっと雰囲気を変えて，この世界にもすっかり慣れたライムちゃんたちは，ますます元気に妖怪退治に励むのだ。

すでに発売中の第7話では，バースが激辛カレーに挑戦したことから始まる大事件。なんと，度重なるバースの悪逆無道に耐えかねた胃袋がついに反乱，妖怪となって飛び出していく。どうする，ライム？

第8話の舞台はマジックショー。ここでも巻き起こる騒動に，美しき女マジシャンに変身したライムちゃん。「ギロチンの練習」なんて怖いことといってるけど，彼女だったら許しちゃう？ところがどっこい。敵もなかなか手強いぞ。危うし，ライム！

ところで，ライムとバースって仲がいいのか悪いのか。最近ちょっとムムムでなんだか気になるぞ。

X68000用　3.5/5″2HD版　各1,500円(税込)
ブラザー工業(TAKERU)　☎052(824)2493

画面はPC-9801版です

発売中のソフト

★DoubleBookin'　計測技研
X68000用　5″2HD版　12,800円(税別)

★宝魔ハンターライム7
ブラザー工業(TAKERU)　2/10
X68000用　3.5/5″2HD版　1,500円(税込)

★卒業～GRADUATION　ブラザー工業(TAKERU)
X68000用　5″2HD版　9.800円(税別)
(TAKERU版)3.5/5″2HD版　7.800円(税別)

★B-FIELD!　ブラザー工業(TAKERU)　2/28
X68000用　3.5/5″2HD版　3,900円(税込)

★デジタルアートコレクション vol.1～8
CONNECTLINE
X68000用　3.5/5″2HD版　各1,500円(税別)
ブラザー工業(TAKERU)　各1,200円(税込)
vol.8のみ　1,400円(税込)

新作情報

★エキサイティングアワー／出世大相撲
電波新聞社　2/25
X68000用　5″2HD版　5,300円(税別)

★宝魔ハンターライム8
ブラザー工業(TAKERU)　3/10
X68000用　3.5/5″2HD版　1,500円(税込)

★ジオグラフシール　エグザクト　3/12
X68000用　5″2HD版　9,800円(税別)

★宝魔ハンターライム9
ブラザー工業(TAKERU)　4/10
X68000用　3.5/5″2HD版　1,500円(税込)

★麻雀航海記　ブラザー工業(TAKERU)　4/未
X68000用　3.5/5″2HD版　5,800円(税込)

★宝魔ハンターライム10
ブラザー工業(TAKERU)　5/10
X68000用　3.5/5″2HD版　1,500円(税込)

★EGWord SX-68K　シャープ
X68000用　3.5/5″2HD版　価格未定

★SX-WINDOW開発キットWorkroom SX-68K
シャープ
X68000用　3.5/5″2HD版　価格未定

★SX-WINDOW開発キット用サポートツール集
シャープ
X68000用　3.5/5″2HD版　価格未定

★マージャンクエスト　SPS
X68000用　5″2HD版　価格未定

★ロボスポーツ　イマジニア
X68000用　5″2HD版　価格未定

★Traum　象スタジオ
X68000用　5″2HD版　価格未定

★鮫！ 鮫！ 鮫！　KANEKO
X68000用　5″2HD版　価格未定

★達人　KANEKO
X68000用　5″2HD版　価格未定

★エアバスター　KANEKO
X68000用　5″2HD版　価格未定

★サバッシュII　ポプコムソフト/グローディア
X68000用　5″2HD版　価格未定

★麻雀悟空・天竺への道　シャノアール
X68000用　5″2HD版　9,800円(税別)

★スタークルーザーII　アルシスソフトウェア
X68000用　5″2HD版　価格未定

★ぷよぷよ　SPS
X68000用　5″2HD版　価格未定

★魔法大作戦　EAビクター
X68000用　5″2HD版　価格未定

★レッスルエンジェルス2
ブラザー工業(TAKERU)　3/未
X68000用　3.5/5″2HD版　価格未定

★あすか120%　ファミリーソフト　3/未
X68000用　5″2HD版　価格未定

★スーパーリアル麻雀IV　ビング　4/未
X68000用　3.5/5″2HD版　12,800円(税別)

★龍虎の拳　魔法株式会社　5/未
X68000用　5″2HD版　価格未定

★餓狼伝説SPECIAL　魔法株式会社　7/未
X68000用　5″2HD版　価格未定

花も実もある高校教師なのだ

他機種では大きな話題になっていたゲームジャンル「教育シミュレーション」。とうとうX68000にも登場です。管理とか指導なんていうとなんか堅苦しそうだけど，相手は5人の女子高生。えこひいきしないで可愛がってあげてね。

Kiyose Eisuke
清瀬 栄介

ジャパンホームビデオの「卒業」がいよいよX68000に登場した。ガイナックスの「プリンセスメーカー」と並んで，子育て型シミュレーションというジャンルを支えている作品である。ただ，このジャンルはいまのところこの2つのメーカーのものしかないんだけどね。

ひとりの娘を6年間かけて育てる「プリンセスメーカー」に対して，「卒業」では女子高生5人の面倒を1年間みる。自分の教育によってみんながどんな進路を歩むことになったかを見るのが，このゲームの楽しみ方だ。

「プリンセスメーカー」では真のエンディングを見たボクも，「卒業」をプレイするのは初めてだ。はたして高校教師の学園生活とはいかなるものなのか？　持田真樹や桜井幸子はいるのであろうかっ!?

4月だよ全員集合

校長「……というわけで，あなたのクラスで問題になりそうな5人をピックアップしておきました」

教師「は，どうも」

受け取ったファイルをめくる。

高城麗子：良家のお嬢様。勉強も運動もできるほうだが，タカビーな性格で教師のいうことに耳を貸さないのが難点。

新井聖美：不良というわけではないが，気の強い娘でこれまた教師のいうことを聞かない。品位に欠け，学力にも問題がある。

志村まみ：まわりの男子高のアイドル的存在。性格は素直だが，気分屋が災いして成績がよくない。

加藤美夏：体育が得意で，学級委員も務める優等生。学力に特に秀でたところがないのと，魅力がないのが難点。

中本静：図書館によくいそうなタイプ。成績は優秀だが，体力がない。人間的な魅力にもいまひとつ欠ける。

……桜井幸子への道は遠そうだ。

教師「わかりました。必ずや彼女たちをこの清華学園にふさわしい生徒に育ててみせます。こう見えてもこのテの名前の生徒の指導は得意なんです」

校長「お，そういえば先生のお名前は」

教師「いかりや長介と申します」

校長「……」

いかりや「ところで，進路を決めるのに魅力や品位が問題になるのですか？」

校長「いや，最近では大学も面接を重視しますのでね。女性としての総合的な人格教育が問われてくるのですよ」

いかりや「女性としての。うふふふ」

校長「ただし！　生徒と仲よくされるのは結構ですが，いき過ぎのないように」

いっとくけど，みなさん！

このゲーム，女の子は脱がないからね。

X68000用　5"2HD版 9,800円(税別)
(TAKERU版)3.5/5"2HD版 7,800円(税込)
ブラザー工業(TAKERU) ☎052(824)2493

仲良くなろうぜ1学期

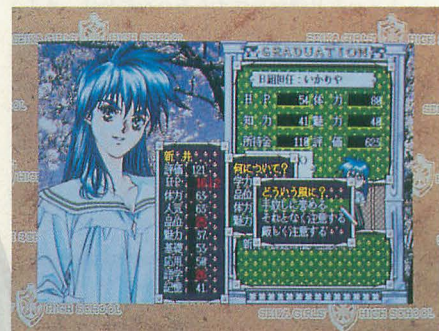
ババンババンバンバン。

いかりや「宿題やったかー」

ババンババンバンバン。

いかりや「課題決めろよー」

この「卒業」では，5人の生徒が1日にやることはあらかじめ自分で決めてある。今日は生け花，明日は問題集といった具合だ。先生であるプレイヤーは，このなかからひとりだけ選んで課題を変えることができる。

この1度にひとりしか干渉できないというのがポイント。いくらこっちが体操で体力を鍛えたいと思っても，ほかの生徒を見ているあいだに勉強ばっかりされたら元のもくあみ。ましてやいいかげんな指示を出していると，時は過ぎるが成果がさっぱりなんてことになってしまう。しっかりしたビジョンと，いつも5人をまんべんなくケアするマメさが重要なんだな。

さぁ，どこからどう教育するかな

子育てシミュレーションのお約束として，これらの1日やったことの成果は生徒のパラメータの増減となって返ってくる。生徒のパラメータは大ざっぱにいって次の通りだ。

・**人気度**：プレイヤーである先生をどれだけ信頼しているか。
・**体力**：生徒がどれだけハードなスケジュールをこなせるかに関わってくる。
・**基礎/語学/記憶/応用**：要は学力。
・**魅力/品位**：人格面のパラメータ。

1年たって，この5人のパラメータがど

これが3年B組5人娘なのだ

のようになっているかによって各自の進路が決まる。生徒が自分で課題を決めているので，自分は同じような方針で指導したつもりでも，やるたびにエンディングが違うことが多い。目指すは５人そろって一流大学合格だ！

さて，では高城くん。課題を見せてみなさい。えーなになに，カラオケ，ドライブ，夜遊び。

……たーかーぎー！

こちらから指示できる課題は主に３系統。作文/暗記/問題集といった，学力を上げるもの。エアロビ/生け花/体操といった，人格面や体力を鍛えるもの。それから個人面談だ。

高城に新井などの教師のいうことを聞かない生徒に対しては，個人面談で信頼関係を少しずつ築いていく。学力の低い志村には作文などの課題を多めに出す。加藤には魅力のつくエアロビをやらせ，体の弱い中本にはまず体操で体力作りだ。

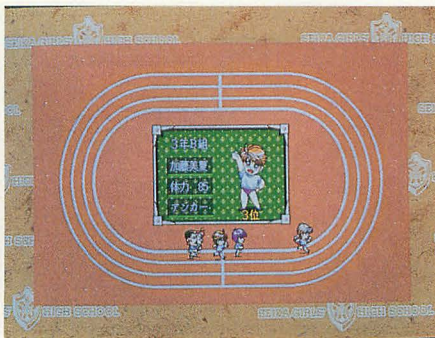
といっても，実際にはなかなか思うようにいかないのがツライところ。面談の予定をスッポかされたり，体力が落ちているのをうっかり見逃すと病気で休んでしまったりする。ま，１学期は生徒の特徴をつかみ，信頼関係を築くのが目的だと思って，気楽にいこう。

学校を舞台にしているだけあって，いくつかのイベントがプレイヤーを待っている。１学期では４月28日に清華祭，６月９日にクラスマッチがある。

清華祭ではクラス対抗演劇コンテストがある。出し物と主役を選ぶと，実際にショートコントを画面上でやってくれる。

中本「鏡よ，世界で一番美しいのは誰？」
新井「それは，それは，森に住む白雪姫」
高城「オホホホホホ。私が世界一よ！」

クラスマッチでは500Mリレーをやる。生徒の体力の鍛え方が試されるわけだ。これまたホントに画面上にトラックを描き，２頭身の生徒がテケテケ走る。そんなに凝った画面でもないんだけど，教え子だと思う

中本！ そんな娘に育てた覚えは……(ぷるぷる)

残念。秋の体育祭こそ優勝だ！

とついつい熱中して見てしまう。入賞するとやっぱり嬉しい。うまい演出だぜ。

どちらのイベントも，いい成績を修めると生徒の評価がアップするので，ただのお遊びではないぞ。これらを通じて，プレイヤーもだんだん生徒に親近感を抱くようになるってわけだね。

夏休みは使いよう

じゃーん。次に控えるは，中間テストと期末テスト。今度は普段どれだけ学力を鍛えてきたかが問われるのだ。

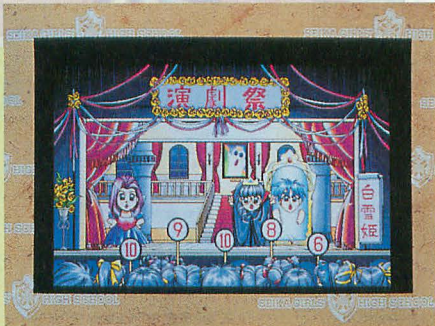
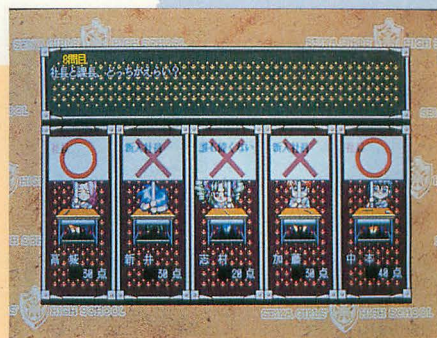
これまた見せ方がうまい。クイズ番組の形式なのだ。

「問題。金は天下の○○○○○」
新井「さらし者」
志村「まわり寿司」

トホホ。新井と志村の低レベル対決には心熱くなるものがあるな。

試験も無事に終われば夏休み。だが教師に休息はない。夏休みの仕事は生活指導。１週間に１人，誰かをマークして，イケナイことをしてないかどうかチェックするのである。

アルバイトをしているとか，海に行くぐらいはカワイイもんだが(先生も海までついて行ったのか？)，飲酒に喫煙とか，夜の街をフラつくとなるとゆく末が心配だ。

そういうところを発見したときに何がいえるかが，教師のウデの見せどころ。生徒の目線で話をすると，人気度が上がることもある。ボクは１学期になつかなかった高

白雪姫が大ウケ。高城は女王のほうがお似合い？

恐怖の中間試験。赤点続出か？

グラフィックに合わせて生徒がしゃべるぞ

城を集中的に生活指導して，ガッチリ信頼関係を築いてしまった。

しかし，夏休み中毎日つけまわして生徒を手なづけようとする教師というのもヒキョーな存在だねぇ。

進路指導にマジになる

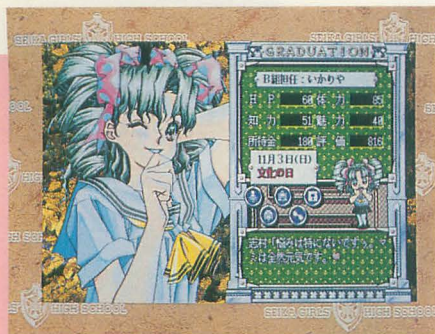
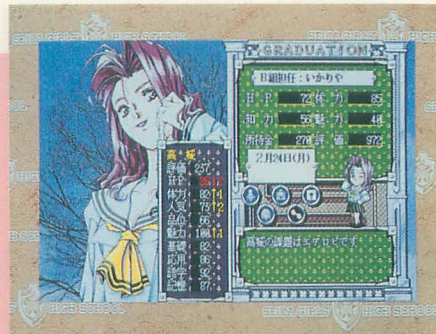
２学期になると，だいたい生徒の特徴も扱いのコツもわかってくる。パラメータも得意なところはより得意に，悪いところもそれなりになってくる時期だ。

そうなると気になるのが，彼女たちの今後の進路。休日の「進路指導」コマンドで，見込みを調べてみよう。

ふむふむ。高城と中本は二流大学，加藤は短大にはいけそうだ。問題はいまのところ専門学校どまりの新井と志村かぁ。

ここで生徒指導のコツをひとつ。５人の生徒のあいだには相性があって，相性のいい生徒と同じことをやらせるとパラメータの上がり方がグンとよくなる。

新井みたいなヒネた生徒が「生け花って，結構面白いんだな」なんて殊勝な発言をするときは，たいてい一緒に生け花をしている生徒がいる。オトモダチに影響されやすいところがいかにも女子高生だね。これがうまく利用できれば，効率よくパラメータを上げることができるぞ。

卒業が近づくにつれて，彼女たちの精神状態も不安定になってくる。ちょっと根をつめると，すぐにノイローゼや病気になりやすい。理由もなく不機嫌になる生徒もい

志村がちょっと色気づいてます

受験間近。高城は余裕のエアロビです

る。こうなると課題をやっても効果が上がらなくなり，パラメータが下がる一方なんてこともある。

特に教師にとって痛いのが失恋だ。妙に魅力のパラメータが高いなと思ったら恋愛中の証拠。卒業までハッピーならいいけど，失恋すると家出をして見つからなくなったり，下手すると妊娠して退学になることもあるらしい。まだボクは未体験だけど。

これらの，悩んだりハッピーだったり，色気づいたり怒ったりというのは，パラメータだけでなくグラフィックでも表される。簡単なことだけど，結構これが「卒業」にハマるのに大きな役割を果たしている。誰だって毎日ふさいでる顔を見れば，なんとかしてやりたくなるもんね。特にタカビーで通っていた高城が初めてアニメーションで笑ってくれたときは，「やっと打ち解けたなぁ」という感じですごく印象的だった。

さぁ，そして3月1日。指導の成果がわかる日がくる。新井と志村には休日のたびに補習をやって，2学期のテストでは全員60点以上取ることができた。体育祭の500Mリレーで優勝もした。秋の文化祭の演劇も好評だった。非のうちどころのない生徒たちだ。進路指導では全員が大学に行けるということだったが，はたして……？

たとえばこんなエンディング

志村「セーンセ」
いかりや「は？」
志村「あたしと結婚できて嬉しい？」
いかりや「う。……そ，そりゃ嬉しいに決まってるさ」
志村「あたしもー！　でもねセンセー，やっぱりあたしと結婚するんだから車は外車じゃなくちゃダメだヨ。あと，家は山手線の沿線に3LDK以上のマンションを買って……」
いかりや「……」

志村と仲よくしたつもりは全然ないのに，この結果はいったい何？

教師いかりやは思わず天を仰いだ。

いかりや「ダメだこりゃ」

*　　　*

いやー，ほかの4人は順当に進学したのだが，志村のエンディングに強烈なのが出てしまった。どの学校に進むかだけがエンディングじゃなかったのね。

なんでも，エンディングは1人につき20通りはあるとかで，結婚なんてのはあまり見られないんだろうけど，それにしても，なんで志村と……？

演出の勝利だね

いや，こいつは想像以上にハマった。基本的には単調な作業の繰り返しなんだけど，5人が次々とトラブルを持ち込んでくるのと，クラスマッチやテストなどのイベントがあることによって，ゲームにすごく抑揚がついている。そのときどきの問題をクリアしているうちに，いつの間にかもう卒業かという感じ。

1年間という期間も，実際にプレイするとトータル3時間弱ってところで長すぎず短すぎず，いいボリュームだ。プレイ時間が長すぎないことと，5人を完全に管理できないので1回1回のプレイに違った展開があるおかげで，何度でも遊べるようになっている。

だけど，大きな視点からひとことだけ。「シムシティ」や「パワーモンガー」さえ昔の作品となったこの時代に，いまだにパラメータを上げたり下げたりを前面に出した「エモーショナル・シミュレーションゲーム」(パッケージより)を作っていていいのだろうかという気がするのだ。

体力は「過労」状態になる30ぎりぎりまで消耗させたほうが効率がいい，なんてことに頭を使うのは，これを買う人が期待している「卒業」というゲームの楽しさとは違うんじゃないかと思うんだよね。

たとえば生徒のひとりにほかの生徒のことを聞いたり，生徒のリアクションに幅を持たせることで，その生徒の性格や状態をつかんだり，生徒同士の相性を知ったりすることはできるはず。いまパラメータになっているものをテキストにすると，プレイヤーは頭のなかで「あ，彼女がこういってるから，きっとこうなんだ」というふうに考えるはず。そうなると「このパラメータはいま37か」と考えるよりも，やってることは同じでも，ずっと教師の疑似体験に近いことができると思うのだ。まぁ，これはこの「卒業」自体のデキとはあまり関係ない話だけどね。

この子育て型シミュレーションゲーム，特にX68000にはいままでなかったジャンルのゲームなので，ぜひとも一度試してみることをお勧めする。PC-9801版の発売から1年半が経っているけれど，古びた印象は全然ない。

でもブラザー工業さん，続編が誕生したあかつきには，早く遊びたいからX68000にもすぐに移植してねん。

負けるな中本，次こそは……

先生の気持ちがわかるかも

生徒をパラメータ化して，変化を見るという非常に単純なゲームだが，前述したようにゲームに変化がつけてあることと，気合いの入ったグラフィックによって，とても楽しめるゲームになっている。

プレイしながら「あいつはテストはできるけど基礎が甘いから受験が心配だなー」なんて自分でドキッとするほどリアルな独り言が出てしまった。やっぱりよくできたゲームなんだろう。

移植の出来は十分合格点。10MHz，メモリ2MバイトのX68000でもストレスなくプレイできる。ただしディスク7枚組なので，ハードディスクはあったほうがいい。アニメのあとのハードディスクのアクセスがちと長いが，サクサク遊べるぞ。

総合評価	0 ― 5 ― 10
操作性	★★★★★★★
グラフィック	★★★★★★★★★
サウンド	★★★★★★★
ゲームバランス	★★★★★★★★
教師実体験度	★★★★★★★★
熱中度	★★★★★★★★★

クイズ正解は勇者のしるし？

Sudo Yoshimasa
須藤 芳政

最近はやりのアクションゲームにハマりすぎて頭まで筋肉になっちゃったそこのオニイさん，いいもんありまっせ。知性を取り戻すにはコレだよコレ。心もなごむすごろくマップで，やわらかコミカルのクイズゲーム。ほら，どーだ。

クイズというものは正解時の快感を味わうもので，不正解がたて続けに出ると，解答者は解答意欲をなくして首をうなだれてしまうものだ。

皆さんも小学生時代に，友達とクイズの出しっこをしたことがあるだろう。幼い心は傷つきやすく，不正解を連発したために「えー!? ○○君こんなこともわからないのー？」といわれ「もう帰る！」といい残して目を潤ませながら家路についてしまった。こんな経験をした方も少なくないでしょう。私もいいたい。「別にそんなこと知らなくたって死なないよーだ！」

よく「～チビッコ日本一」とかいう題目で司会者の出題に得意げな顔で答えているようなお子様。お兄さんは大嫌いだぞ～。

出発するのよ

この物語の舞台はビーフィールドという魔法王国。「そこって，練馬の近くかな？」と思った人，全然違いますよー。

ある日，ビーフィールドの王妃が王を石に変えて国宝である幻帝の鏡を持ち去ってしまう。国は国宝を取り返してくれる勇気ある者を募ったが，旅立った者たちは皆帰って来なかった。

「センセー！」

魔法学校の教師であるディルズが学校の営業不振に浮かない顔をしているところへ，教え子のレイディが元気よく飛び込んできた。幻帝の鏡を取り戻せば，学校の人気も

お爺さんだけじゃなくお助け兎も出てくる

上がってハッピーに暮らせるとディルズに訴える。「このままミジメな生活を続けるよりはマシ」ということで国宝奪還を決意した彼らは，手始めとしてとりあえずレステルの森にいる予言師のもとへと向かうのだ。

しかし，いままで誰も帰って来ないというのに，そんなことあっさり決断してしまっていいのか若者よ。私だったら家でよだれ垂らして寝てたほうがいいけどなぁ。

選択式でよかった

プレイヤーは，ディルズとレイディのどちらか1人を選んですごろく形式のマップ上を移動。邪魔する者たちをクイズに正解することで打ちのめし，ゴール地点へたどり着くことが目的。

ピロロロロ……。

スロットマシンのドラムのようなものを回転させて進むコマ数を決定する。出る数は1～4だが，4はなかなか出ない。

おっと，いよいよクイズの出題だ。答えは4つのなかから1つを選ぶのだが，間違えたり制限時間内に答えられなかったりすると体力が減ってしまう（レベルアップで体力値の上限が上がっていく）。体力が0になるとゲームオーバーになるので，危なくなったら宿屋で体力を回復しよう。アイテムショップには，使うと選択肢が2つに減る道具など便利なものがある。

歩いている途中でときどき現れるお爺さんは4枚のカードを差し出してくるので，1枚めくらなければならない。「出た目だけ戻る」「出た目だけ進む」「ワープ」「スタート地点へ戻る」のどれかになるのだが，数歩引き返すならともかく，スタート地点へ戻るのは辛い。「わーい，もうすぐゴールだよ～ん」なんていうときにスタート地点行きは勘弁願いたいぞ！

あと，ディルズはジャンル選択ができるのに対して，レイディはノンセクションに固定となっているらしい。「それならディルズのほうがいいや」とも思えるが，会話の内容はそれぞれ別なので，ぜひ両方でやりましょう。レイディのほうが会話がブッ飛んでいて面白いと思うぞ。

わからん！

はっきりいって，私にはちょっと苦しい問題だらけ。たとえばタイトルも知らない小説の登場人物名なんてわかるわけない。ある種の本を読破している人なら楽勝か。地理や歴史もとっくの昔に忘れてしまっていて解答意欲消失寸前。でも不正解のとき，ディルズが「だー！」っていってたり，レイディがハンカチ噛んで悔しがっている顔が笑える。

対戦モードもあるので，友達を打ちのめして優越感に浸ろう！　問題暗記すりゃ楽勝楽勝一！　一度やりだすと結構，熱中してしまうかも。

X68000用　3.5/5"2HD版　3,900円(税込)
ブラザー工業(TAKERU)　☎052(824)2493

ヨロコビの声も欲しいな

PC-9801からの移植という形だが，曲の演奏はZ-MUSICを使用して，ドラムをAD PCMで演奏している。98からの移植ものにありがちな「パスゥ」という腑抜けたドラムサウンドを聴かないで済むのだ。

ついでだから，正解したときには「やったー！」，不正解時には「だー！」とかしゃべらせたほうが面白かったかもしれない。

総合評価 (0～10)	
ハマり度	★★★★★★★
音楽	★★★★
ギャグ絵	★★★★★★★★

唸る鉄拳が魂を斬る！

簡単操作で敵を打ち砕く爽快感がたまらない「マッドストーカーX68」。製品版ではvsCPUモードもついて，より完成度が上がっている。皆さんにも，ぜひ，この美しいロボットアクションを体感してもらいたい。

Taki Yasushi
瀧　康史

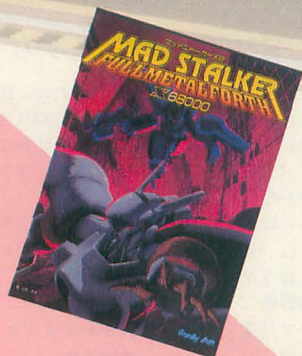
2月号で紹介した「マッドストーカーX68」だが，新たに届いた製品版を見てびっくり。グラフィック周りはもちろん，新しくvsCPUモード，技の相殺という要素も加わり，かなり別のゲームとなってしまっていた。そこで，急遽もう1回追加レビューとして製品版をもとに「マッドストーカーX68」を紹介していく。

そして，今月は紹介というよりも，より強くなるための攻略ということで話を進めていくことにする。先月号との重複はできるだけ避けるつもりなので，先月号をちゃんと読んでおいてから今月号を見ていただきたい。

弱肉強食チャート

細かいところがビシバシ変わったので，チャートにも違いが現れた。技の相殺という要素が加わったため，絶対的な対空技がキャラクターによってはほとんどなくなってしまったからである。

先月号のチャートとの違いで目立つ部分といえば，まず，ハウンドドッグの上昇とシルフィードの弱体化だろう。このあたりがどういう理由で起きているかを中心に話を進めてみる。

なお，技の名前はすべて奥義書にあるものを利用しているので，わからないものがあったら奥義書を参考にしていただきたい。

X68000用 5″2HD 2枚組　7,800円（税別）
ファミリーソフト　☎03(3924)5435

待ち神威を垂直2段ジャンプでかわし……

小キックで降りて，反撃開始！

とりあえず，前回掲載したものとの違いを見てほしい。

神威vsハウンドドッグ

まず，ハウンドドッグの攻撃のバリエーションであるスプレッドショットは神威には使えない。これは，神威のつむじ撃ちの強が，移動中完全無敵のためである。

また，オーバーブーストなどは，神威の張扇斬りなどによって，簡単に返されてしまう。神威は突進型の技である，つむじ撃ち，紫電断烈斬が強いため，地上では割と簡単にハウンドドッグに対してイニシアチブが取れる。

しかし，神威は一度転んだ場合，ハウンドドッグの跳び込み攻撃が非常に苦手となる。神威には上段，下段当て身投げ（先月号にはなかった）があるので，跳び込みに関して，絶対的な防御を誇るかに見えるが，実はそうではない。神威の当て身投げは，上段からの攻撃を防御した場合，防御したところが投げの間合いより広いときには，当て身投げにはならないからだ（ただし，技は完全に相殺される）。したがって，ハウンドドッグが，長い足を利用して足の先が神威の頭ギリギリ届く位置で攻撃したら，イニシアチブはハウンドドッグにまわってしまう。

また，神威には小の紫電断烈斬が強力な対空技になるのだが，技の相殺を狙った攻撃をハウンドドッグにされると，神威には選択の余地があまりなくなってしまう。まして，2段ジャンプをされたら……。

したがって，一見神威が有利に見えるこの組み合わせも，5分5分ということにした。ライジングドッグvs神威の場合も似たような理由で，5分5分にしてある。

シルフィードvsハウンドドッグ

シルフィードは未知数の要素をもったキャラクターであるが，現状ではハウンドドッグのいいカモになってしまう。

シルフィードが，ハウンドドッグを苦手とする最大のポイントは，対空攻撃の貧弱さである。

神威の紫電断烈斬の弱と同じ要領で，最初の攻撃判定が出るまでは無敵なことは確かなのだが，この短い間をきっちり合わせるテクを身につけても，ハウンドドッグに最初の出始めの攻撃を相殺されてしまい(相殺されると一瞬止まり，お互いノーダメ

しゃがみ小パンチをキャンセルして……

ージのまま進行する)，動きの素早いハウンドドッグは，次にローキックなどが簡単に入ってしまう（ハウンドドッグのローキックは，連続技への布石である)。まして，2段ジャンプをハウンドドッグが利用したら，さらにいやなことになる。

一方，制空権が完全に取れそうなシルフィードだが，ハウンドドッグは2段ジャンプをもつため，シルフィードより高くジャンプし，攻撃を一方的に加えることができる。つまり，空中でもほぼ互角になってしまう。このあたりが，類似キャラであるライジングドッグvsシルフィードが5分5分なのに，ハウンドドッグの場合は7分3分になってしまうのだ。

ライジングドッグorハウンドドッグvsゴング

ゴングがライジングドッグに弱いのは，まず信頼できるはずの対空攻撃，ブーメランクラッシュが，簡単にライジングドッグのニードルキックで返されてしまうこと。また，黄金パターンである，飛び道具を撃たせて落とす方法が，見事に使えることなどがあげられる。

ハウンドドッグの場合，ブーメランクラッシュを使えば比較的楽に落とせる黄金パターンが完全ではない（スプレッドショットを放ったあとバックする)。そのためライジングドッグvsゴング，ハウンドドッグvsゴングには，差をつけさせてもらった。

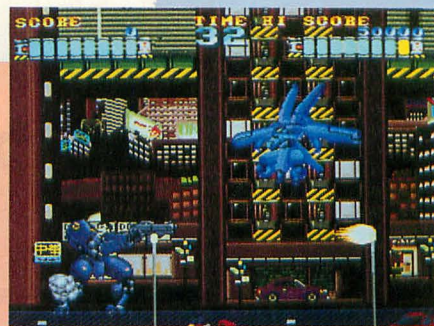
ゴングvsその他

ゴングは攻撃力は弱いが，スピードは非常に速い。したがって，スピードで翻弄して少しずつダメージを与えていくのが勝つための定石となる。

ゴングが苦手とする，ハウンドドッグ，ライジングドッグは，飛び道具をもっている。つまり，ローリングクラッシュで攻撃する場合は，相手の隙をつかないと防御されてしまう。

また，ハウンドドッグ，ライジングドッ

すぐに吸い込んでしまえ

弱体化が目立つシルフィードに明日はあるのか

シルフィードに飛び道具を使うのが悪いのさ

グには，完全な対空攻撃がある。これによって，ブーメランフラッシュが防御されやすい。さらに，ドッグ2体は動きが比較的速く，ゴングの動きにある程度ついてくるのだ。ここまでくるとゴングが有利なはずはない。

一方，神威，プリソナーβ，シルフィードには，完全な対空攻撃がなく，ブーメランクラッシュがなかなか返せない。神威はまだ自分自身が速く動けるからいいが（つむじ撃ち)，プリソナーβ，シルフィードは隅に押しやられると，逃げることさえ難しい。

なお，ゴングにはかなりダークな技がある。それは，ドロップキックからのダッシュ1本背負いである。間になにか技をはさんでも最後をキャンセルすると，なにもできないことが多い。また，起き上がり無敵技がない場合，起き上がりにドロップキックを重ねられると，防御しても食らっても，ドロップキックをキャンセルで強の1本背負いを出されてしまい，まず間違いなく投げられてしまう。

いろいろ試したあげく，完全にハマるとはいえないが，これは限りなくハメに近いと思われたので禁じ技ということにしておいた。チャートはこの技を使用せずに作られているので，ゴングはこの地位に留まってる。

もっとも，ゴングはドロップキックをキャンセルして，ブーメランクラッシュを出すだけで，プリソナーβと，シルフィードには，優位に立てるのだが。

プリソナーβvsその他

プリソナーβには明示されてはいないが，実はひそかに，神威より強い当て身投げが存在する。

それは，グランドバキュームだ。

グランドバキュームは，出始めが一瞬無敵っぽい。当たり判定がない無敵ではなく，当たることは当たるが，必ず相手の技を相殺してしまう変わった技となっている。

しかも，お互い技を相殺し合ったあとは，プリソナーβには，吸い込みの最中なので，まず間違いなく吸い込める。つまり，当て身投げどころか，上段，中段，下段関係なく，すべて身抱き締め（延長可能）ができる。

神威と違い，多少広い間合いでやっても，吸い込める場合が多いため，どんなときでも使える（神威には自分の真上以外は使わないほうがいい)。

全体的に動きは遅いのだが攻撃力があるため，ほとんどすべてのキャラクターと同等に戦える。しかし，唯一ゴングのローリングクラッシュは，当たっても跳ね返ってしまうため，当て身投げ狙いはほとんど失敗するので注意。

また，動きの遅さが災いして，ゴングに翻弄されがちになってしまい，どうしてもイニシアチブを取られる。

燃えるアクションゲームだ

遊んでみるとやはり面白い。これにつきるね。読者のみんなもゲーセン仲間でも集めて，自宅でわいわいやってみなさい。ハマりまくるから。さらに，罰ゲームで一気なんて用意した日には（納豆＋蜂蜜＋味噌汁がお勧め)，もう，超燃え，燃え燃えっす。

あと，製品版では，vsCPUモードもついているので練習はいつでもできるだろう。ほかのキャラでシナリオを解くモードは？って，それはどうやら，㊙であるらしい(私にもわかってません)。わかった人はぜひご一報を。

表　弱肉強食チャート

	ハ	ラ	神	ゴ	プ	シ	Total
ハ	＼	5	5	6	5	7	28
ラ	5	＼	5	7	5	5	27
神	5	5	＼	5	5	5	25
ゴ	4	3	5	＼	6	6	24
プ	5	5	5	4	＼	5	24
シ	3	5	5	4	5	＼	22

チャート作成協力：斉藤昭夫，白井五三雄

※2月号に掲載したタイトル写真で登録商標がソフトウエア工業となっていましたが，正しくは光画堂スタジオが登録商標を所有しています。また，製品版と大幅に違うサンプル版で評価してしまったことで，関係者各位にご迷惑をおかけしましたことをお詫びいたします。

特別編1

いやらしいケンになる！

KEN

Taki Yasushi 瀧 康史

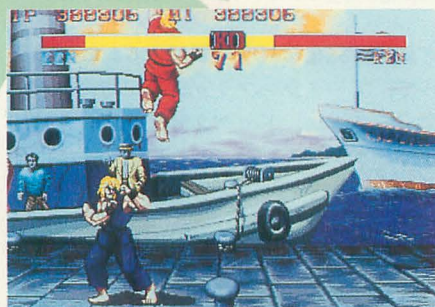
ハッキリいおう。ダッシュのケンは強くない。むしろ弱い。そのケンでいかにして勝つか？ これは難しい。理想としては，

「ケーン，きちゃだめー」

とリンのように対戦相手にいわれるほど強いケンになりたい。どうすればいいか？ 簡単ではないが方法はある。いやらしいケンになればいいのだ！ ケンにはダイナミックさがあるが，そのほとんどは裏目に出ている。私も含めてプレイヤーもダイナマイトな性格の人が多いが，そこはそれ，ケンでのプレイを心に決めたらストIIは「ランス」シリーズよりもエッチなゲーム！(私はダークな紺のケンが好き)

定石の戦法&連続技

跳ばせて落とすのが基本。すなわち波動拳で跳ばせて昇龍拳で落とす。ただしケンの昇龍拳は，リュウの昇龍拳に比べて出がけの1000点時間は少ない。当たった瞬間ちょっと重くなる感じがするのが1000点昇龍拳だが，ケン使いは昇龍拳のためにケンを選んだようなものだから，やはり必ず1000点を狙わねばならない。「1000点昇龍拳以外は昇龍拳とは呼ばない！」これである。

ケンはリュウよりも跳び込み大パンチが強い。具体的なことは後述するとして，この大パンチを利用していかに攻撃を結びつけるかを考えよう。オーソドックスなのは大パンチ，中足払い(キャンセル)，波動拳。これは必ずしも当てるためではないと考えてもいい。当たればそれでいいが，もし当たらなくても相手を隅に追い詰めることができる。追い詰めたら，波動拳＋昇龍拳で，

落ちてきたら小キックから同時押し連続技

いわゆる鳥籠ができるしね。そう，波動拳の戻りグラフィックはキャンセルできるぞ。

最も強力な攻撃力のある連続技は，立ち小キック，しゃがみ小キック，立ち小キック＋大パンチの同時押し(キャンセル)昇龍拳。最後の同時押しは大パンチのグラフィックだが，実は，グラフィックとダメージ以外の当たり判定，キャンセル(できるかどうか)は，立ち小キックとして扱われる。同様に中パンチとかも同時押しで使える。これは，コンシューマ機ではPCエンジンとメガドライブにしかできない(が，PCエンジンはちょっとタイミングが違う気がする)。スーパーファミコンでは，同時押しではないが同じ技ができる。

ただ，この技はザンギエフなどの大きめのキャラにしか最後の昇龍拳が決まらない。小さい敵には最初の小キックを出さないで小足払いからすればよい。敵が小さくてぴよぴよのときは，素直に跳び込み大パンチ，大アッパー小昇龍拳だ。

これらの必殺技は，それはもう，隙さえあればバシバシ入れるべきだが，具体的にいえば，敵が昇龍拳などの上昇技から無防

近すぎず，遠すぎず，間合いを保つ

1000点昇龍拳以外は昇龍拳じゃない!!

サマーソルトは大パンチでぶっつぶせ！

備に落ちてきたり，ぴよぴよになっているとき，敵がめくろうとしたときに自分が潜れた場合など。うまく全部決まればぴよらせることができるので，もう1回入れよう。

当たり判定，食らい判定

真にいやらしいケンになるためには，連続技だけ覚えても意味がない。難しいコマンドだって，慣れれば機械のように誰でもできるからだ。同時押しだって，最初は難しいが慣れれば勝手に手が動く。

方法としてはフェイントもあるが，これは，読み合いだ。波動拳を撃つふりをして撃たない，などバリエーションはいろいろ。しかし，これだけではまだ足りない。

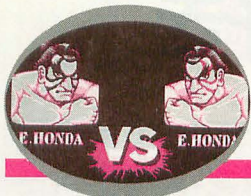
では，どうすれば，敵が嫌がるケンになれるか？ それは，当たり判定と食らい判定を意識したプレイをすることである。

一般技で強い攻撃というのは，ダメージが大きいだけではなく，当たり判定と食らい判定の差が大きいことがある。

具体的にいえば，ケンの跳び込み大パンチは，当たり判定(敵に攻撃を加える判定)は手よりひとまわりかふたまわり大きいぐらいにあるが，食らい判定は手首より奥の前腕伸筋の辺りにしかない。つまり，ケンの手首は「無敵」といえる。キックも同様で足首は無敵。すなわち，うまくめり込ませれば，サマーソルトも蓋をするように返せるのだ。これを狙わない手はない。

ケンでは，この当たり判定と食らい判定の範囲を意識するのが必勝法だ。

まさか起き上がりに小キックを重ねて同時押し技へ，なんて考えてる人はいないよな？ そんなことしたら起き上がり昇龍拳の餌食だぜ！

特別編 2

ああっ本田様っ！ そのクマドリの下にあなたの素顔を見た！

Komura Satoshi 古村　聡

E.HONDA

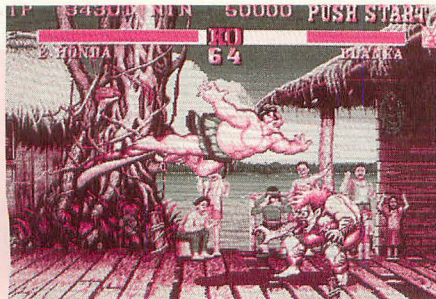
本田様は偉大なのだ。その証拠にストIIで「様」をつけて呼んでいいキャラは本田様だけなのだ。なぜそういうことになったのかわからないが私がそう決めたんだから間違いないのだ。ちなみに「餓狼伝説」シリーズではクラウザー様とギース様も「様」をつけていいことになっているのだが，あの２人は「会長」「社長」と呼ばれることもあるから，やっぱり本田様がいちばんエライのだ。文句のあるヤツぁ前に出ろ！

あふれる愛ですべてを制す

偉大な本田様にいちばん似合う対戦キャラ。それはなんといっても百裂キックの脚線美中華娘・春麗なのだな。対戦前の顔なんかどう見たって美女と野獣だ。どうでもいいけど，アーケード版のスーパーストIIの本田様は怖いクセに妙に間抜け顔で，まるでいつもいいとこまでいきながら自分のドジで悪だくみに失敗するアメコミのチンピラ悪役みたいになってしまっている。私は悲しい。

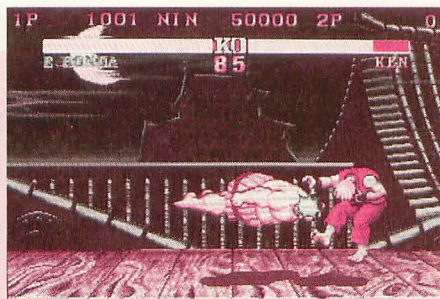
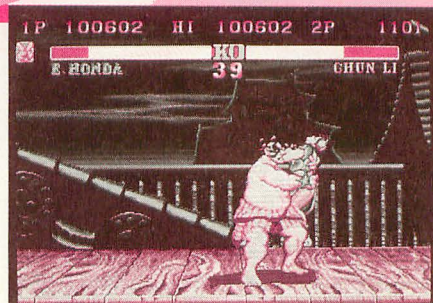
しかしだ。本田様は本当は色男なのである。その証拠をお見せしよう。春麗に近づいて……そーれ，→＋大パンチ！

お前も我が輩の愛のトリコになるのだ。わはははは！　抱きしめ抱きしめ！

って一感じでこのサバ折り，別名愛の抱擁をする。どーだ色男だろう。もしそのとき，春麗が愛の力でフラフラになっていたら(世間ではピヨるともいうらしい)ついでにそのまま投げ技で寝かしてしまえ！……ちなみにこの技，相手が春麗のときは愛の抱擁で，男ならサバ折りと私は呼んでいる。男なんかサバで十分だ(ブランカと愛を確かめ合うのもそれはそれで……やっぱりヤだ)。

このサバ折りには意外な効果もある。春麗使いが春麗を使うわけは，たいていは春麗が好き，ってことだ。だから本田様の愛の抱擁によってプレイヤーはエナジーを吸いとられてフラフラ状態になってしまうので，簡単に倒せるのだ。……怒らせちゃって逆効果の可能性もあるけど。

なんかこう書いてると本田様の戦いって肉弾戦っていうよりは神経戦って気がしない？　しないか……。

先月号の中野氏の２ボタンスティックでの攻略では，春麗は引きつけて頭突き，ということになっているけど，６ボタンが使えるなら，個人的には叩き落とし(というのか？　要するに立ち大パンチ)のほうがいいような気もする。タイミングがちょっと難しいけど。あと，フライングスモウプレスも結構効く。

アーケード版ではIIからII'になってバランスにいろいろと変更が加えられた。本田様の場合，例の叩き落としはちょっとタイミングがシビアになったけど，代わりにスーパー頭突きは出やすくなった。また，百裂張り手をしながら前進後退できるようにもなった。実は私はII'の本田様は歴代のストIIのなかでも最もバランスのいいキャラなのではないかと思っているのだ(Turbo/スパIIでは追加されたスーパー百貫がいまイチ使い道がないのでくやしい)。あとは飛び道具があれば……。あ，当て身投げもほしいかな。

今日も我が愛のトリコ作りに励むのだ！

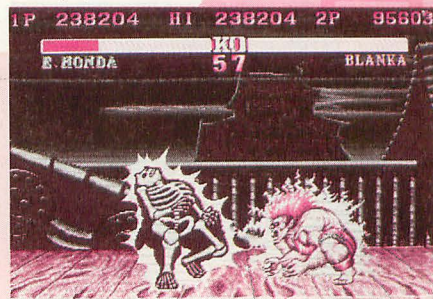
派手さの陰には孤独の悲哀？

そういえばアーケード版ではブランカの電撃を受けたときに，マゲのなかにも骨が見えたことから「本田はマゲのなかにフライドチキンの骨を隠している」っていわれてたけど，X68000版で見ると，なんだか頭蓋骨からマゲの骨が枝分かれしてるように見えるんだよね。X68000版の本田様はきっと生まれながらにして相撲取りになる運命だったんだろうなぁ。一子相伝の怪しげな相撲技を伝授されてたりして。あっ，スーパー頭突きがそうなのか？

それにしても，考えてみると本田様の風体ってすごいよなー。マゲにクマドリ(歌舞伎役者なんかが顔に塗ってるアレ)，シマシマのデカパン。本物の相撲取りはクマドリなんてしてないぞー。若ノ花も武双山も琴錦もしてないぞ。っていうかスポーツ選手でクマドリをしてるのなんてプロレスのザ・グレートカブキくらいのもんだし。

ところで本田様の正体は，相撲界を追われ，プロレス界にもいられなくなった双○黒(○尾)のなれの果てだって噂なんだけど，ホント？

フライングスモウプレスは結構効くぞ

一子相伝の怪しげな技!?　スーパー頭突き

マゲの骨は我が宿命の証！　かもしれない

特別編3

やっぱり赤いザンギが好き

ZANGIEF

Sudo Yoshimasa 須藤 芳政

ダッシュ時代に私が最も使いまくったキャラがザンギエフ(以下ザンギ)だった。

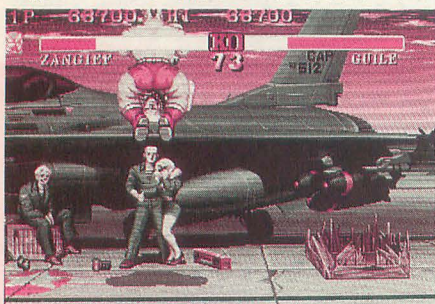
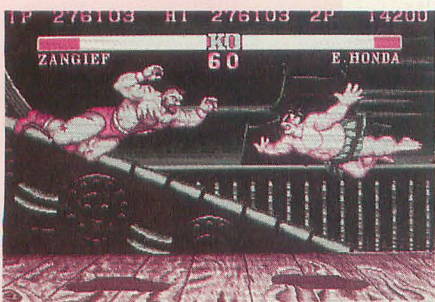
なにしろスクリューパイルドライバー(以下スクリュー)を3回キメるだけで問答無用に相手をKOなのだから,「こりゃお得ですがな」と私は欲の皮をつっぱらせて,せっせとスクリューの吸い込み練習に励んだものだ。

まず初めに覚えたのが,しゃがみ小パンチ+しゃがみ小キック+スクリュー。その実践例は,相手がダウンしたところへ密着し,ひたすらしゃがみ小パンチを連打しておいて一発,相手がガードした瞬間に小キックを入れて吸い込むという何とも卑劣な技だった。そして前述の3つを入れる前へ

「好きでーす」

とでもいいたげに跳び込む,めくりボディプレスをプラスして,私は悪に染まっていった。しかし「密着」になるチャンスがあまり訪れないため,

「せめてお近づきになりたいわー!」

ということで今度は立ち小キック+スクリューをマスター。これならある程度間合いが離れていても可能なので,起き上がりに投げられる心配もない。私はこれで対戦相手をハメまくり,端っこへ追い詰められたバイソンなど私にとってはクリスマスを前にした七面鳥に等しかった。

技をくらっても無表情なガイル

夜空を舞い,見つめ合うふたり

男として見損なったぞザンギ!

しかし! 「起き上がり昇龍拳野郎」の増殖によって私の闘いにも限界が見え始めた。ダウンした相手に小キックを入れた次の瞬間には私のザンギちゃんはマヌケな面で宙を舞っているという光景を,頻繁に目にすることになったのだ。

「んもう!こんなことって許せる?」

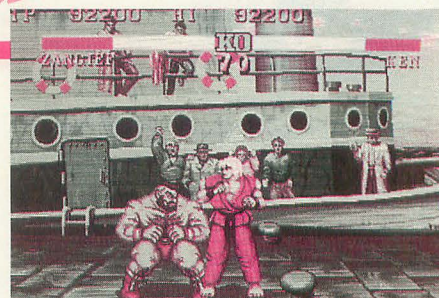
と,私はいままで行ってきた卑怯な行為を棚に上げて悔しがり,今度は立ちスクリューを覚えることにした。

「いままではパンチやキックを入れたあとでしか吸い込めなかったから,負けてたんだよねー」

おっと,男は言い訳なんかしないんだぜ。

根性&努力&多大な練習時間で立ちスクリューをマスターし,相手が跳び込みや昇龍拳をスカったあとにも吸い込めるようになって思った。

「ああ,学校のレポート早くやらなきゃいけなかったんだ……」

ところで,ザンギでプレイしてて乱入されるといちばん嫌なのがサガットだ。近づけないので嫌なんだな。サガット君~,格闘技にはスキンシップというものが必要なのだよ。そこんところを君はわかってないねー。なんだね,あの「アイガー! アイガー!」っていうのは! 上か下か白黒つけなさいっての! まったく。だけどそれでいて,自分もときにはサガットを使っていたりする。人の心っていうのは複雑なものなのだ。

試合中にガマンできなくなったか?

そのほかの技についてもちょっと書こうか。前にも述べたボディプレス,私は「好きですアタック」と勝手に呼んでいる。本田もスモウプレスなんてワザを使えるが,あっちのほうが両手を広げていて「愛」が感じられるところがちょっと悔しい。あと,ゲーム中に出そうともしていないのに「腹つかみ」が出てしまうことがある。これは春麗にやると完全なワイセツ行為ではないだろうか(考えすぎ?)。

そういえば,ダッシュのザンギは赤と緑の2種類から選択するが,どうもあの緑色のやつは使う気がしなかった。なぜ傷口が緑なのだ? 勝って「ぐわっはっはっは」と笑っている姿を見ると,口の中まで緑色ではないか! 傷口には緑の絵の具を塗って(緑の絵の具って毒だって聞いたことがあるけど……),試合中に噛み砕いたホウレン草を舌の上に載せて見せびらかしたとでもいうのか? そんな下品なヤツは好かん! というわけで赤を私は使っていたのに,ダッシュターボ時代になってからあの「脂の乗ったザンギ」に乗り換えた。この浮気者めが!

＊　＊　＊

あーあ,こんなことばっかり書いちゃって……。「何か技について参考になること書いてるのかなー?」なんて読んだ人は怒り心頭かも。ごめんなさい。

「フウッ!」(「フンッ!」という人もいるが私は「フウッ!」だと思う)「グルグル……」「ズデーン!」という行程で決まるスクリュー。あんな高さで頭から落とされたら頭蓋骨骨折は必至なのに,みんなピンピンして「昇龍拳!」とか叫んでいる。そんな石頭な君たちが大好きだよー!

特別編4

ハメれば君もヨロレイヒー

BALROG

Nishikawa Zenji 西川 善司

飛び道具がない。グラフィックパターンがケチられていてパンチボタン押してもキックが出たりする。本拠地スペインでは金網に登れて結構得意気だけれど，跳び降りてくるところを敵に必殺技発射準備完了状態で待たれちゃって結局おマヌケ。バック転も彼ならではの特殊技，一瞬は無敵。けれど解けて体制を整え直すときに無防備で完全カモ状態。ちょっと連続技を食らうとすぐピヨる。

なんだか見かけのかっこよさの割には，いまいち冴えない彼。でもほかの11人とはまったく違う要素をたくさん持っている彼。だから私はバル彦さんが好き。スペイン人なのにヨーデルを歌う彼が好き。勝利したときにはバック転までして喜ぶリスのような無邪気さが好き。

突け！刺せ！削れ！

バル彦さんの特徴はなんといってもその腕につけた爪。これがあるときとないときでは戦闘能力値は雲泥の差だ。敵の攻撃をガードしすぎて爪が壊れ落ちるまで，バル彦さんの爪の恐ろしさを相手に思い知らせよう。

まず第1にしゃがんだ状態での爪による牽制攻撃，これが基本。弱は連射が効き，強は連射性はないけれど足払いを食らいにくい。ただ，やみくもに連射するんじゃなくて相手が間合いに入ったと思ったらズビャ！って感じで。「おい，それなしにしようぜ！」って対戦相手にいわれたら「ヒョー」とかいってごまかせ。

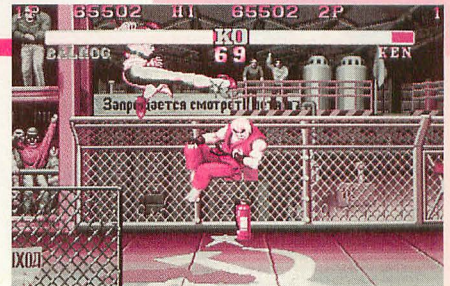
相手が跳び込んできたら，立ち中爪で刺すか，垂直ジャンプキックで蹴り落とせ。この2つを覚えれば君はもう仮面のナルシストだ。

次は必殺技のゴロゴロの使い方だ。相手を転ばせたあとゴロゴロを重ねて削りたい。けれど，相手が起き上がり技(昇龍拳，サマーソルトキック)で返してくるような場合は禁物。けれど積極的に使いたい。善司流バル彦さんは，後ろに下がったり後退ジャンプを織り交ぜて突然小ゴロを出して相手に奇襲をかける。これは対戦で使える。必殺技をかけるために間合いを詰めてくる相手に，突然ゴロ，ズビャ！　朝起きての放尿より気持ちいい。バル彦さんのゴロゴロは大中小で回数が違うが，これはいい替えれば前進する距離が違うということ。だから，奇襲をかける場合にも相手が遠いときは大や中で，近ければ小でと使い分ける。

以上をマスターすれば，地上での闘いで不利ということはない。もう，君はヨロレイヒーの叫びが喉元でウズいているはずだ。

ハメて何が悪い，べらぼーめ

世の中にハメ技禁止という軟弱なルールがあるそうだが，ゲームの普通の遊び方で繰り出すことができる技ならばどう組み合わせようがプレイヤーの自由だ。負けてイヤならゲームはするな。というわけで，バル彦さんには強力な連続技がある。

まず，相手にジャンプ中キックなどで跳び込む。この跳び込みは着地したときに相手に密着するくらいの近いところから。もちろんうまくやらないと返り討ちにあう。ここで相手がガードしてくれたらもう80%成立したも同然。相手のガードを確認したらレバーを横に入れて中または大パンチを連射だ。相手をそのままレインボースープレックスで投げ潰すことができる。

ジャンプ攻撃には垂直跳び蹴りで

これは相手の起き上がりに重ねてもいい。もちろん返されることを計算して戦略的に使わないと駄目だが，相手を画面端に追い詰めてやると成功率は格段に上昇。

これをマスターすればいわゆる不利な敵キャラももう怖くない。叫べ！　ブルーライトヨロレイヒー。

ヒョーは勝利の合い言葉

バル彦はなんといってもヒョーだ。ヒョーのないバル彦なんて湯切りをしないでラ王を食べるみたいなものだ。

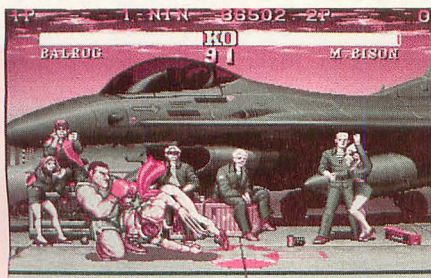
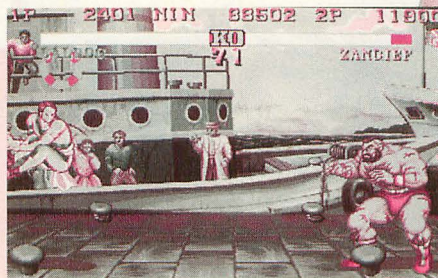
バルセロナアタック/イズナドロップは壁を後ろに背負って出せば，壁を蹴って，すぐに相手のところへ跳んでいける。また，この壁を蹴ったあとの軌道はジョイスティックで制御できるので，相手に向かうと見せかけてフェイントで着地。爪攻撃，ズビャなんてこともできる。

イズナドロップは相手と離れてても決まるので，この間合いを自分のものにしろ。慣れれば平気でブラ男の電撃も投げられるようになる。

君がグレープなら僕はグレープフルーツ

もう誌面が尽きた。バル彦さんは実は奥が深い。まだまだ隠されたSUPER DEATHな技があるのに紹介できないとは，こりゃ善さん一生の不覚だね，ぺちぺち。

対戦してくれる相手がいるならバルーバル対戦が熱い。ぜひ一度お試しあれ。

それでは，みなさんブルーライトヨロレイヒー(ホントはなんていってんだ？)。

ゴロゴロで削れ

壁を背にして，バルセロナ/イズナ

特別編5

魅惑の春麗至上主義

Kiyose Eisuke 清瀬 栄介

もう，誰がなんといおうと春麗である。春麗以上に闘う姿の美しいキャラクターはいない！　後追い格闘ゲームには春麗モドキのキャラがたくさん出たけど，どれも春麗の足元にも及ばないね。フフン。

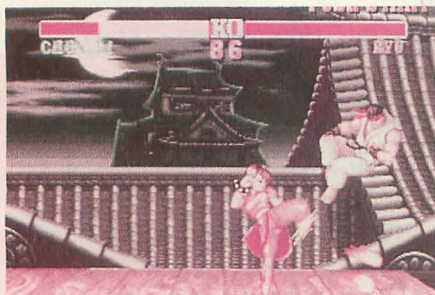
ボクは元祖ストII以来ずっと春麗がメインキャラである。もうキャラとの相性の問題など関係ない。負けたら，もっと強い春麗になって相手を倒す！　これが愛情ある正しい春麗使いの姿というものである。

だが，春麗には飛び道具がなく，ストIIのキャラの中では立場は弱い。かなりうまい春麗使いでも，波動昇龍拳だけのお子様リュウケンにハメられることもある。

しかし！　それぐらいで春麗をあきらめるようでは愛情が足りん。通常技を駆使し頭を使って，お子様リュウケンに本当の強さは必殺技乱発ではないということを，身をもって知らせてやらねばならない。動きの速い春麗は，工夫をすれば相当強い。それに，春麗で勝つためには相手の心理を読むことがすごく大事なので，奥が深いキャラクターだともいえる。

春麗3段講座

春麗の必殺技はあんまり頼りにならないが，ひとつだけ春麗を強力にサポートしてくれる技がある。

いわゆる「跳び込み3段」がそれだ。跳び込み大パンチ+掌底+百裂キックの連続技である。すごいぞー，これは。1回決まれば相手はピヨってしまうので，自動的に2回決められる。合計すると9割がたの体力を奪えるのだ。

百裂キックでガリガリ削れ

元祖ストIIの登場以来，強い春麗のスタイルというのも移り変わってきたが，この春麗3段はダッシュもスーパーも変わらず強い。X68000で練習したあとは，ゲームセンターでキャミィ相手に決めることもできるというわけ。春麗が好きなら絶対にマスターしてほしい。

跳び込み3段ができない人のために，主な症状と対処法を紹介しよう。

1）跳び込み大パンチのあとに返される

パンチのタイミングが早くて，掌底と連続技になっていない。最初の大パンチは着地直前までガマンしよう。

2）掌底が出ずに百裂キックになる

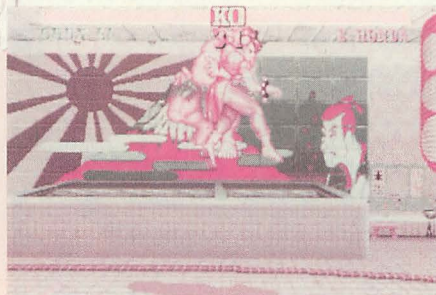
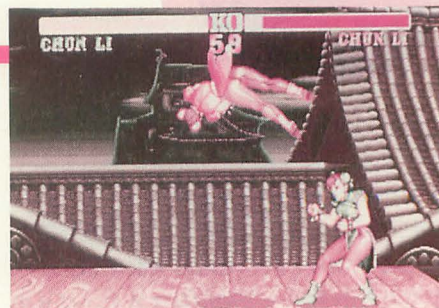
大パンチをちゃんと押してない。落ち着いてパンパンと大パンチを叩き，それから中キック連射でも間に合う。

3）百裂キックが当たらない

相手を追いつめていると最後の百裂が当たりにくくなる。画面の端に追いつめたときは掌底のあとキックや投げ技を使う。

もうこれさえマスターすれば昇龍拳なんかうらやましくない。実は最近マスターしたのでボクも浮かれ気味なのだ。うりゃ，ガスガスガス。美しい……（バルログかお前は）。

多彩な攻めで昇りつめる

春麗には飛び道具がないので，ただ待っていても勝てない。かといってただ跳び込んでも昇龍拳の餌食になるのがオチ。相手に行動を起こさせて，それにうまく反応しないと勝ちは見えてこない。1つひとつの例を挙げていたらきりがないので，使える

美しい空中投げをマスターしよう

春麗3段2連発でここまで減らせる

技を紹介しよう。

●踏みつけ：ザンギエフやサガットなど，大きいキャラクターにイヤがられる。近いところから跳び上がり，すぐにレバー下+中キックを連打すると，ガンガンヒットしながら春麗が相手の体を昇っていくぞ。

●投げ：春麗やるなら絶対投げを使え。昇龍拳，タイガーアッパーカット，サマーソルトのあとはもちろん，相手の起き上がりざまに百裂キックと見せかけて投げるのだ。相手が警戒してガードしているところを投げちゃうなど応用範囲は広い。

相手を跳び越して投げてしまうめくり投げも基本だから覚えておこう。さらに美しさにこだわるなら，竜巻旋風脚投げと，イズナドロップ投げの2大空中投げはマスターしておきたい。

●跳び込み：普通は小キックで跳び込むが，相手の攻撃ももらいやすい。当てたいときだけ中キックを出すのがおすすめ。

●百裂キック：起き上がりざまに当てて，相手の体力をケズる。スーパー頭突きやローリングアタックにも効く。

●中脚払い：リーチが長い。波動拳やソニックブームを封じられる。

無敵技の少ない春麗は，自分の技について熟知していることがより重要になる。このキックの当たり判定は，出ている時間は長いかなど。慣れれば春麗でもガンガン攻め込んでプレッシャーをかける闘い方ができるようになるはずだ。その境地に達してもリュウケンとは五分五分までいけるかどうかってところだけど（悲しい）。

ところで，どうして春麗は中国人なのに「やったー」と叫ぶんだろう？　「スピニングバードキック」は？　はて？

【特別企画】

ひなまつりPRO-68K

Oh!Xでの付録ディスクも6つ目。そろそろ転換期にきています。ディスクは誌面とは違ったメディアです。可能性は尽きません。これまでも「フロッピーディスクメディア1枚でどれだけのことができるか」ということについては，それなりの主張を持って付録ディスクを作成してきました。それにより犠牲にするものが大きすぎるということもわかりました。
今回は逆に詰め込まない方針で作成したのですが，それでも予想よりは詰め込んだかたちになってしまったようです。
前回のディスクでは未完成なものが多すぎるという批判もあったのですが，本誌上では「作品」として完成されたプログラムが提供されることはほとんどありません。これは過去の例を見ても明らかでしょう。「素材」であるか「道具」であるか，あるいは，もっと漠然とした「指針」のようなものを提供しています。
ですから「付録ディスクがつくとプログラム入力がなくなって楽になる」という考え方はかなり根本的なところで間違っています。提供されるものが多い分だけ，暗黙のうちに要求されるものも多くなっていることも忘れないでください。

CONTENTS

付録ディスク使用上の注意

収録プログラム&データの使い方

編集部

それではさっそく付録ディスク「ひなまつりPRO-68K」に収録されているプログラム&データについて解説していきましょう。ディスクを展開する前に必ず目を通しておいてくださいね。

お待たせしました。最初に予告した時期からかなり遅れてしまいましたがようやく付録ディスクをつけることができました。

めでたく3月ということで,「ひなまつりPRO-68K(特に深い意味はない)」をお届けします。

毎度のことながら付録ディスクに収録されたプログラムはLHA.X(吉崎栄泰,岡田紀夫)で圧縮されていますので,ツール類を使用するためにはあらかじめ展開作業が必要になります。

付録ディスクの展開について

付録ディスクは展開作業により2HDディスク3枚分に展開されます。事前にフォーマットしておいた2HDディスク3枚を用意しておいてください。

では,展開作業に入ります。

付録ディスクをドライブ0に入れOPT.1キーを押しながらリセットしてください。あとは画面上に表示されるメッセージに従ってどのディスクを展開するかを選択していくだけです。付録ディスクに収録されているプログラムとデータ(展開後のもの)を表1に示します。指示に従ってディスクを入れ換えていってください。

今回のシステムでは自動起動のディスクは作成されませんから,各自がお使いのシステムを立ち上げて,それぞれのディスクの中身を見ていってください。

展開したいものを選ぶ

各ディスクの内容

●ディスク1

Morph! (柴田 淳)

以前より本誌上で作成していたモーフィングツールの完成版です。2枚のグラフィック画像を滑らかに補間して表示することができます。もともとアニメーション用途のツールですので,できあがった画像を再生するためにはDōGA CGAシステムのツールが必要となります。

AMIシステム (福嶋章太)

このツールを使うとMOドライブなどのSCSI装置から高速にデータを読み込みVRAMに展開することでアニメーションの連続再生を行うことができます。

サンプル画像の生成プログラムやサポートツールも付属しています。

カードゲーム支援関数CSF (高山忠信)

複雑になりがちな多人数版のカードゲームをシステマティックに管理して簡単に記述できるようにするためのCARD2.FNC用の高位関数をまとめたライブラリです。カードをスプライト表示するCARDSP.FNCとサンプルゲーム2種も収録されています。

●ディスク2

SX-WINDOW関係のツールをまとめたディスクです。いずれも実行環境としてはSX-WINDOWver.3.0が必要ですので注意してください。

SX-BASIC (石上達也)

お待たせしました。以前より本誌上で開発中とアナウンスされていたSX-WINDOW上で動作するBASICインタプリタと支援システムです。

現在の段階ではまだ暫定版ということで,不安定な面や仕様上の不備もありますが,公開デバッグの意味も含めて発表ということになりました。試用してみて不都合な点や拡張してほしい点などがありましたらぜひ編集部までご連絡ください。このシステムを使った投稿などもお待ちしています。

なにぶん完成品ではありませんので,試用する際には編集中の文書の保存などは念入りにやっておいてください。

MTEXT.X (田村健人)

シャーペン.Xで採用されている拡張テキストフォーマットの文書を表示,簡易エディットするためのツールです。シャーペン.Xで作成されたものなら,ビットマップデータを使ったものからEasydrawデータを張り込んだものまでほぼ完全に再現することができます。

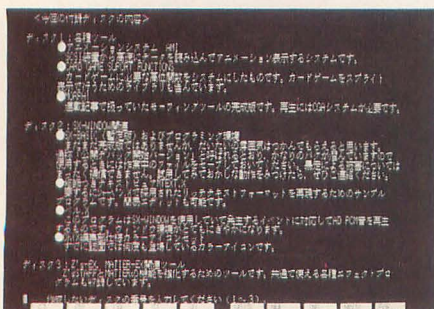
ベル.X (石田伯仁)

SX-WINDOWで発生するさまざまなイベントに対応させて音声を出力させるプログラムです。ウィンドウ環境がいちだんと賑やかになります。ディスク容量の都合上,サンプル音声は4種類だけですが,簡単に手持ちのデータを追加できますので各自で拡張してみてください。

カラーアイコン集 (編集部)

Oh!X編集部で使用されているSX-WINDOWのICON.LBです(標準のものに色を乗せただけという話もあります)。そのまま使用することもできますが,場合によっては設定を多少変更することが必要になると思います。念のために元のICON.LBは保存しておいてください。

すでに自分でかなりアイコンを書き換えてしまっているという人はICUP.X(田村健人)を駆使してみてください。

SX-WINDOWver.3.0のパッケージに

ついてきた標準のICON.LBをICON0，付録ディスクに収録のものをICON1，現在あなたが使用中のものをICON2ののようにリネームしてカレントディレクトリに集めておき，

ICUP ICON0 ICON1 ICON2

のように実行します。すると，ICON0とICON1の差分をICON2に加えてくれます。アイコンが重複して登録されているときはどちらを採用するかを聞いてくるので適当なものを選択していってください。なお，新しいファイルはICON.LBというファイル名でカレントディレクトリに作成されるので注意が必要です。

なお，このツールで書き換えられるのは#128以上のPAT3とPAT4のみです。シンボルやメニューには関与しません。

●ディスク3

Z's-EXおよびMATIER-EX，その他，共用の外部フィルタプログラムを集めたディスクです。

Z's-EX&MATIER-EX　（菊地　功）

X68000用の代表的グラフィックツール，Z'sSTAFF PRO-68KとMATIERの機能を拡充するためのツールです。アナログマスクや変形ツール，画像フィルタなどの機能が組み込めます。専用システムコールの採用とライブラリの整備により，外部プログラムの作成が非常に簡単になりました。もちろん，ユーザーが作成したプログラムも簡単に登録できます。

Z-MUSICシステム追補データ

昨年末に発売された「Z-MUSICシステムver.2.0」で収録漏れになってしまっていたデータとツールをまとめたものです。ご迷惑をおかけしました。

表1　展開後のファイル内容

```
DISK 1
¥
|
|-morph!
|   |-makefile
|   |-morph!.c
|   |-mosub.c
|   |-mosub2.c
|   |-mosub3.c
|   |-mpat.c
|   |-trtrfm.c
|   |-MOHEAD.H
|   |-compl.s
|   |-fnt_put.s
|   |-trtrfm2.s
|   |-txfill.s
|   |-txline.s
|   |-txt_ps.s
|   |-txt_put.s
|   |-Morph!.x
|   |-otog.pol
|   |-ostritch.pic
|   |-giraffe.pic
|
|-card
|   |
|   |-csf
|   |   |-CSF.bas
|   |   |-CSF_c.bas
|   |   |-CSF_SP.bas
|   |   |-CSF_SP_c.bas
|   |   |-CSF.doc
|   |   |-CSF_vr.doc
|   |   |-CSF_fr.doc
|   |   |-CSF_far.doc
|   |
|   |-card
|   |   |-old_maid.bas
|   |   |-old_maid.x
|   |   |-old_maid.doc
|   |   |-whist.bas
|   |   |-whist.doc
|   |   |-whist.x
|   |
|   |-cardsp
|   |   |-CARD_SP.doc
|   |   |-CARD_SP.FNC
|   |   |-card_sp.s
|   |   |-CardSpLib.s
|   |   |-CardSpLib.l
|   |   |-sample.bas
|   |   |-sample2.bas
|   |   |-CardSpLib.o
|   |-CARDDRV.x
|   |-TR.DAT
|   |-CARDLIB.A
|   |-readme.doc
|
|-ami
|   |-AMI.DOC
|   |-BOXinBOX.x
|   |-AMIENV.x
|   |-AMIPC.x
|   |-AMIZM.x
|   |-AMI.x
|   |-AMIFE.x
|   |-BOXinBOX.c
|   |-toG3R3B2.x
|   |-AMILIB.H
|   |-toTONE16.x
|   |-WAVE.x
|   |-WAVE2.x
|   |-AMIENV.s
|   |-AMIPC.s
|   |-AMIZM.s
|   |-makefile
|   |-AMIFILE.s
|   |-AMIPLAY.s
|   |-AMILIB.MAC
|   |-AMILIB.l
|   |-AMI.s
|   |-AMIFRAME.s
|   |-AMIFE.s
|   |-G3R3B2.pal
|   |-toG3R3B2.has
|   |-TONE16.pal
|   |-toTONE16.has
|   |-G2T_4.pal
|   |-MONO.pal
|   |-WAVE.c
|   |-WAVE2.c
|   |-AMIFE.DOC

DISK 2
¥
|
|-sxbasic
|   |
|   |-sample
|   |   |-dentaku.sxb
|   |   |-sample.sxb
|   |   |-TEST.sxb
|   |   |-ファイル.sxb
|   |   |-INPUT.sxb
|   |   |-配列.sxb
|   |   |-switch.sxb
|   |   |-if1.sxb
|   |   |-if2.sxb
|   |   |-b1.PT4
|   |   |-b2.PT4
|   |   |-c1.PT4
|   |   |-b3.PT4
|   |   |-b4.PT4
|   |   |-b5.PT4
|   |   |-btest.sxb
|   |
|   |-engine
|   |   |-engnf.c
|   |   |-engn.c
|   |   |-engine.x
|   |   |-engn.h
|   |   |-engnd.c
|   |   |-engnc.c
|   |   |-makefile
|   |
|   |-edit
|   |   |-TESCROLL.C
|   |   |-TEDIT.C
|   |   |-TEDIT.H
|   |   |-BEDIT.X
|   |   |-TEMAIN.C
|   |
|   |-wind
|   |   |
|   |   |-pat
|   |   |   |-text.bmp
|   |   |   |-text2.bmp
|   |   |   |-bin2pat.c
|   |   |   |-bin2pat.x
|   |   |   |-stn2.bmp
|   |   |   |-stn.bmp
|   |   |   |-cur.bmp
|   |   |   |-cur2.bmp
|   |   |   |-Hvar.bmp
|   |   |   |-Hvar2.bmp
|   |   |   |-select.bmp
|   |   |   |-select2.bmp
|   |   |   |-alt2.bmp
|   |   |   |-alt.bmp
|   |   |   |-updown.bmp
|   |   |   |-updown2.bmp
|   |   |   |-picture2.bmp
|   |   |   |-picture.bmp
|   |   |-makefile
|   |   |-wind.x
|   |   |-windf.c
|   |   |-sdbmp.h
|   |   |-windd.c
|   |   |-winde.c
|   |   |-windc.c
|   |   |-windr.c
|   |   |-windb.c
|   |   |-wind.c
|   |   |-wind.h
|   |-exp.c
|   |-comp.c
|   |-sx.c
|   |-inter.c
|   |-efunc.c
|   |-table.c
|   |-MACHINE.S
|   |-SXBASIC.H
|   |-dialog.c
|   |-LOGO.BMP
|   |-SxBasic.x
|   |-sbfile.c
|   |-sbedit.c
|   |-makefile
|   |-file.c
|   |-suzume4.sxb
|   |-sxbasicc.bfd
|   |-enginc.bfd
|
|-mtext
|   |-Makefile
|   |-draw.c
|   |-et.doc
|   |-key.c
|   |-mtext.c
|   |-mtext.doc
|   |-mtext.h
|   |-mtext.lb
|   |-mtext.x
|   |-save.c
|   |-text.c
|   |-tm 3.h
|   |-wmfore.has
|
|-icon
|   |-Makefile
|   |-icup.c
|   |-icup.x
|   |-icon.lb
|   |-readme.doc
|
|-bell
|   |-sasasa.pcm
|   |-ベル.x
|   |-ベル.DOC
|   |-ベルX.pt4
|   |-ベル.S
|   |-shhhva.pcm
|   |-cha.pcm
|   |-quao.pcm
|   |-ベル.ENV

DISK 3
¥
|
|-zs&mat_ex
|   |
|   |-effect
|   |   |
|   |   |-sample
|   |   |   |-ATOG.C
|   |   |   |-ATOG.x
|   |   |   |-ATOG2.c
|   |   |   |-ATOG2.x
|   |   |   |-REVMASK.c
|   |   |   |-REVMASK.x
|   |   |   |-SOUNDFILER.C
|   |   |   |-SOUNDFILER.x
|   |   |   |-TRIPLE.C
|   |   |   |-TRIPLE.x
|   |   |   |-RESIZE.C
|   |   |   |-RESIZE.x
|   |   |   |-LUPE.c
|   |   |   |-LUPE.x
|   |   |   |-makefile
|   |   |
|   |   |-source
|   |   |   |-picfiler.c
|   |   |   |-xpic.s
|   |   |   |-mask2gram.s
|   |   |   |-alternate.c
|   |   |   |-typedef.h
|   |   |   |-mat3.h
|   |   |   |-perspective.c
|   |   |   |-pers.c
|   |   |   |-mat3.c
|   |   |   |-maskpaint.c
|   |   |   |-maskoff.s
|   |   |   |-monotone.c
|   |   |   |-g_monotone.s
|   |   |   |-fractal.c
|   |   |   |-fractalh.c
|   |   |   |-flare.c
|   |   |   |-differ.c
|   |   |   |-compose.c
|   |   |   |-cutfiler.c
|   |   |   |-GtoMask.s
|   |   |   |-exeyes.c
|   |   |   |-verno.s
|   |   |   |-to2.c
|   |   |   |-palet.c
|   |   |   |-cgrad.c
|   |   |   |-mask.c
|   |   |   |-mask_.s
|   |   |   |-zoomio.c
|   |   |   |-aspect.s
|   |   |   |-exopen.s
|   |   |   |-flare2.c
|   |   |   |-twinkle.c
|   |   |   |-wand.c
|   |   |   |-lupe.s
|   |   |   |-amiex.c
|   |   |   |-exec.c
|   |   |   |-to64k.s
|   |   |   |-extype.c
|   |   |   |-makefile
|   |   |   |-twirl.c
|   |   |   |-shusa.c
|   |   |-picfiler.x
|   |   |-alternate.x
|   |   |-perspective.x
|   |   |-maskpaint.x
|   |   |-monotone.x
|   |   |-fractal.x
|   |   |-fractalh.x
|   |   |-flare.x
|   |   |-differ.x
|   |   |-compose.x
|   |   |-cutfiler.x
|   |   |-GtoMask.x
|   |   |-exeyes.x
|   |   |-verno.x
|   |   |-to2.x
|   |   |-palet.x
|   |   |-cgrad.x
|   |   |-mask.x
|   |   |-zoomio.x
|   |   |-aspect.x
|   |   |-exopen.x
|   |   |-flare2.x
|   |   |-twinkle.x
|   |   |-wand.x
|   |   |-amiex.x
|   |   |-exec.x
|   |   |-extype.x
|   |   |-twirl.x
|   |   |-shusa.x
|   |   |-twirl2.x
|   |
|   |-source
|   |   |-window.c
|   |   |-effect.c
|   |   |-filer.c
|   |   |-zs.c
|   |   |-Zstartup.s
|   |   |-transfer.s
|   |   |-rev.s
|   |   |-Ztrap7.s
|   |   |-MS_LIMIT.S
|   |   |-Zs_EX.Mak
|   |   |-mat.c
|   |   |-Mstartup.s
|   |   |-maskconv.s
|   |   |-Mtrap7.s
|   |   |-Mat_EX.Mak
|   |   |-GPAINT_B.H
|   |   |-GPAINT_B.S
|   |
|   |-lib
|   |   |-EXLIB.DOC
|   |   |-EXLIB.H
|   |   |-EXCALL.MAC
|   |   |-EXCALL.DOC
|   |   |-GETADR.S
|   |   |-CONFIRM.S
|   |   |-SELECT.S
|   |   |-FILEWIN.S
|   |   |-GRAM2BUF.S
|   |   |-BUF2GRAM.S
|   |   |-REVLINE.S
|   |   |-REVBOX.S
|   |   |-REVFILL.S
|   |   |-MCSET.S
|   |   |-ALTERNATE.S
|   |   |-GPAINT.S
|   |   |-GETAREA.S
|   |   |-GETPOINT.S
|   |   |-ROLLUP.S
|   |   |-ROLLDOWN.S
|   |   |-WINITEM.S
|   |   |-OPENWIN.S
|   |   |-MANAGE.S
|   |   |-MOVEWIN.S
|   |   |-CLOSEWIN.S
|   |   |-WINBOX.S
|   |   |-TRANSWIN.S
|   |   |-VERSION.S
|   |   |-SETCOL.S
|   |   |-GETCOL.S
|   |   |-ACTPAL.S
|   |   |-KEYINT.S
|   |   |-MASKADR.S
|   |   |-COMGVRAM.S
|   |   |-BUFFADR.S
|   |   |-EXNUM.S
|   |   |-GPAINT_M.S
|   |   |-EXLIB.MAK
|   |   |-EXLIB.L
|   |   |-MINT.S
|   |   |-LEDIT.S
|   |-Mat_EX.X
|   |-Mat2_EX.X
|   |-zs_ex.sys
|   |-Zs_EX.X
|   |-mat_ex.sys
|
|-omake
|   |-VIP.ZMS
|   |-68SND.ZMS
|   |-M1_ZM_STD.MDD
|   |-BOS.CNF
|   |-ZMD18.R
|   |-COMCHK.R
```

SX-BASIC公開デバッグ第1回

SX-BASICプログラミング環境とは

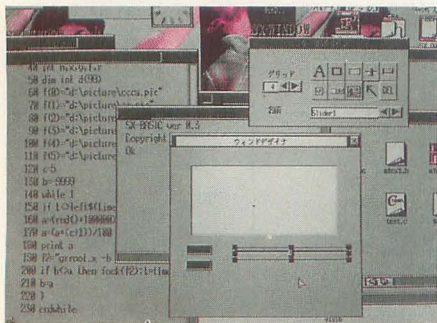
Ishigami Tatsuya 石上 達也

SX-WINDOW上で手軽にプログラムが作成できるようになりました。言語仕様はX-BASICコンパチで簡単,ウィンドウプログラムに関して特別な知識は要求されません。暫定版ですが雰囲気は十分に味わえます。

SX-WINDOWの出現によって,X68000の操作環境は格段に向上しました。

キーボード主体であったコマンドラインの環境から,マウス主体の視覚的な環境へと変化し,より直感的な操作が可能となりました。また,複数のプログラムの同時実行が可能となり,グラフィックエディタで図を作成しながら,テキストエディタで文章を書く,というようなことも可能となりました(つい最近ですが)。

しかし,SX-WINDOWの登場によって,操作が簡単になった半面,そのプログラムを作成するのは,以前にもまして複雑な作業となってしまいました。単独で実行されるコマンドライン上のプログラムと違い,SX-WINDOW上のプログラムを作成するには,それまでになかったような新しい概念を覚えなければなりません。イベントドリブン,メモリハンドル,リソースファイル……。

さらに,具合の悪いことに,現在,SX-WINDOW上で動作するアプリケーションを作成するにはC言語かアセンブラを用いなければなりません。アセンブラはともかく,C言語を用いるにしても,ポインタやハンドル,さらには構造体のリスト構造など,初心者には取っつきにくい分野をひとと おり習得する必要があります。

まずはウィンドウデザイン

SX-BASICシステム

コマンドライン上の環境に比べて,SX-WINDOW上のプログラムでは,扱うデータの種類が格段に増加しています。ウィンドウポインタ,グラフポートポインタ,テキストハンドル……などなど。

このように多種のデータを扱うには,アセンブラのように,すべてのデータ構造を無視してメモリアドレスの問題に還元してしまうか,扱うデータ構造をプログラマが自由に定義できるC言語のようなプログラミング言語を用いなければなりませんでした。

アセンブラを用いる場合,プログラミング言語自体の仕様が,データ構造を無視しているので,プログラマが,常にデータ構造を意識していなければいけないということです。また,C言語にしても,定義されたデータ構造をプログラマが完全に知らなくてもよい,ということではありません。せいぜい,変数の型を合わせてくれるとか,オフセットを自動的に計算してくれるとか,その程度のことです。

ところで,これらのデータ構造のうち,大部分がSXプログラムに特有のデータを扱うのに用いられます。ここでいう,SXプログラムに特有のデータとは,ウィンドウの位置を記憶する変数だったり,テキストの状態(全体の文字数,カーソルの位置,セレクトの範囲など)を記憶する変数だったりします。このようなデータは,SX-WINDOWのガイドラインに従ってプログラムを作成している限り,ほぼ一定の扱いを受けます。たとえば,ウィンドウの移動はウィンドウのタイトル部分を左ドラッグするとか,右ボタンでメニューが出てくる,という具合です。

このような決まりきった処理以外で,SXプログラムに特有なデータというのはほとんど用いません。プログラムの残りの部分を作成する際に使用するのは,ごく一部分です。めったに使わないデータのために,開発言語は,柔軟な,あるいは複雑なデータ構造をサポートしなければならないのです。

SX-BASICでは,この条件を逆手にとって,複雑なデータ構造を扱えないけれども,その代わり,プログラマも複雑なデータ構造を考慮しなくてもよい,という環境を実現しています。

扱えるデータ構造は,X-BASICとまったく同じです。「ウィンドウポインタ」や「テキストハンドル」といったデータ構造はサポートしていません。つまり,プログラマはこれらのデータを意識しなくてもいいのです。

だからといって,これらのデータを必要とする機能(ウィンドウを開くにはウィンドウポインタの意味するところのデータが必要ですし,同様にテキストを扱うにはテキストハンドルの意味するデータが必要です)が実現できないわけではありません。

これらは,SX-BASICのシステムの中で,プログラマが意識しないところで,ひっそりと「普通に」処理されています。

奥まったところにあるウィンドウが左クリックされれば手前に出てきますし,右ボタンが押されればメニューが出てくる,というのがSX-WINDOWでいうところの普通の処理です。

普通に動作していれば,意識しなくてもよいデータ構造はSX-BASICがシステム内で処理していますので,プログラマはまったく関与しなくてもかまいません。つまり,プログラムする必要がなくなり,言語自体がそれらのデータ構造をサポートしな

くても済むわけです。

ただし，SX-WINDOWのガイドラインをあえて破るようなプログラム（左クリックでウィンドウが消える，メニューが左ボタンで出てくるなど）を作る場合には，システムの行っている動作を変更しなければなりません。SX-BASICでは，そのようなことができるだけの豊富なデータ構造をサポートしてないので，不適切です。

このようなプログラムを作る場合には，C言語のように柔軟なデータ構造をサポートした言語か，最初からデータ構造という概念のないアセンブラのような言語を用いてプログラミングを行わなくてはなりません。

システムの概要

このシステムはSX-BASIC，ウィンドウデザイナ，ウィンドウエンジンの3つのプログラムから構成されています。

このうち，前者2つは単独で使用することもできますが，これら3つのプログラムを組み合わせて使用することにより，手軽にSX-WINDOW上のプログラムを作成することができます。

●ウィンドウデザイナ

ウィンドウデザイナはSX-WINDOW上で動作し，対話的にウィンドウの設計を行うことができます。

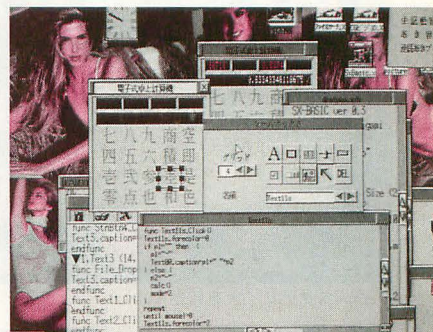
設計はマウスを用いて画面上の座標を指定するので，ユーザーは座標などの数値を気にする必要はありません。また，デザイン中にプロパティ（後述）も指定できるので，ウィンドウの実行時とまったく同じデータを表示/編集することが可能です。

ウィンドウデザイナにより作成されたファイルは，そのままSX-BASICで読み込むことが可能です。

今回の付録ディスクに収録されたバージョンは1993年10月号の「秋祭りPRO-68K」に収録されたバージョンよりも，いくらかデバッグと改良がなされています。基本的な操作方法には変更ありませんが，テキストエディタ（今回の付録ディスクに収録のBEDIT.X)を呼び出すなど新しい機能がいくつか付加されています。詳しくは，次回に説明します。

●SX-BASIC

SX-BASICはSX-WINDOW上で動作するインタプリタ型のBASICです。後述する点を除いて，本体付属のX-BASICと文法的に互換性を持っています。ですから，X-BASICをいままで使ってきたユーザーは違和感なくSX-BASICを導入することができます。

また，SX-WINDOW用のプログラムを作成するのに便利な命令がいくつか拡張されています。

各種プログラムを立ち上げてプログラミング

SX-BASICの仕様を列挙すると，

・数値配列は3次元まで，文字列配列は2次元まで使用可能
・タスク間通信を用いて，他タスクとの通信が可能
・CPUのレジスタを直接制御することにより，より低レベルなプログラムの記述が可能
・Aラインコード，Fラインコードを実行可能
・ウィンドウデザイナの出力するファイルを直接実行することが可能
・特にウィンドウエンジンとは密な通信が可能

となっています。

●ウィンドウエンジン

SX-WINDOWはGUI（Graphical User

SX-BASICでのプログラミングについて

えーと，SX-BASICデバッグ担当の中野です。SX-BASICは，こういってはなんですがよく飛びます。マスターアップ後にわかったことですが，収録されているOBJR型のプログラムではなぜか必要量のワークが確保されません。SXBASIC.XとSXBASICC.BFDをカレントディレクトリに置いて，BUP.X（秋祭りPRO-68Kに収録）で，

```
BUP SXBASICC
```

のようにしてOBJC型のファイルを作成してください（念のために入れておいてよかった)。

ではサンプルの解説からいきましょう。

サンプルプログラムでなけなしの実用性があるのは電卓のみです。ほかはデバッグ用や機能紹介用のテストプログラムです。

●SAMPLE.SXB

ウィンドウデザイナのテスト用に作ったウィンドウにちょこちょこ機能を加えたものです。

起動直後は，ウィンドウをいじるとSX-BASICが受け取っているタスク間通信の内容を表示します。SX-BASICのウィンドウでブレイクキーを押し，実行を中断するとマウス操作に応じた反応がウィンドウに表れるようになります。これを見てもわかるように，SX-BASICのプログラムは普段は動いていません。プログラムは入力されていますが，実行は停止しています。

ウィンドウエンジンからは，普通のBASICでいうところの「ダイレクトモード」でSX-BASICで記述された各関数を実行しているのです。起動時のみWindow設定部の直後に書かれたプログラムを実行しますが，これは初期化動作がほとんどだと思ってかまいません。

このプログラムではスライダーとテキストの動作だけ見てもらえればいいでしょう。プロパティはこういうふうに扱います。

●DENTAKU.SXB

ボタンが多いだけで特に変わったことはやっていません。

使い方ですが，空はCA,即はCEです。最上段はメモリエリアで直前の計算結果を4つまで保存できます。隙間にある白い三角を押してください。表示部を押せばカレント文字列として呼び出せます。

当初の予定では答も日本語化されるはずだったのですが，表示範囲の制限によりあきらめました。さらに16進化も考えたのですがわかりにくいのでやめました（優良可不留放というのも過激だし)。

●BTEST.SXB

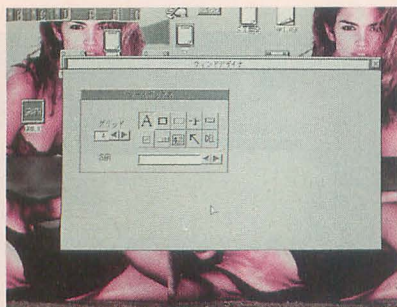
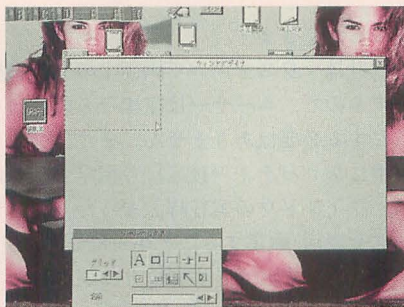
これは理不尽でないスクロールバーをSX-BASICで自作してみたものです。ビットマップのドラッグなどの処理は参考になるかもしれません。

同時にファイルドロップやメニュー表示のサンプルでもあります。

●要注意事項

・コマンドは小文字のみ（省略形はl.r.?のみ）
・大文字，小文字はしっかり区別
・変数宣言はプログラムの先頭でのみ可
・IOCS関連は使わないほうがいいぞ
・文字列長の制限はないが，Textのキャプションは90文字まで（きびしー）
・キー入力はinkey$だが，リアルタイムキー入力に変更されている。これはウィンドウエンジンのウィンドウではなくSX-BASICのウィンドウで受け付けられる（なんじゃこりゃ？）

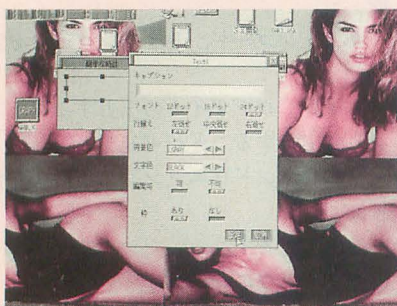
（中野修一）

3分でできるSX-BASICプログラム

ウィンドウデザイナを起動

これくらいの大きさにして……

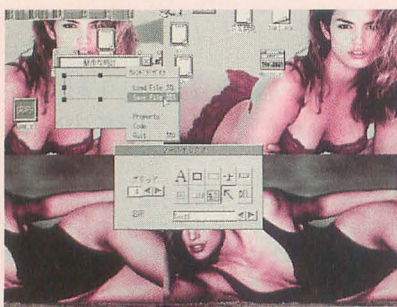
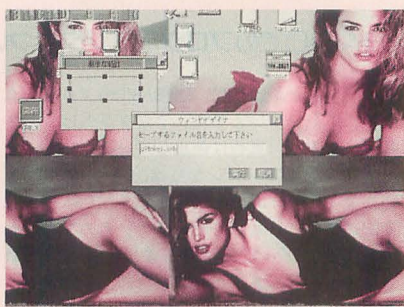
タイトルのメニューでpropertyを選ぶ

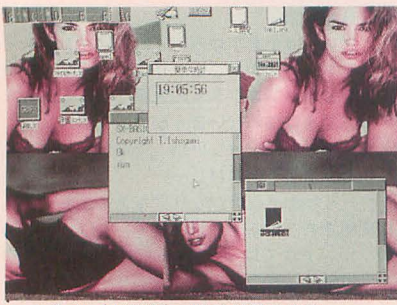
グローボックスを禁止して名前を入れる

「A」のアイコンでテキストエリアを指定

テキストのところでpropertyを選択

24ドット文字，背景をLGRAYに設定

タイトル部のメニューでcodeを選択

プログラムを書き込む

タイトルのメニューでsave fileを選ぶ

ダイアログ内にファイル名を指定する

SXBASIC.Xを立ち上げアイコンを放り込む

runと入力（小文字）。時計が動き出す

SX-BASICシステムのプログラミング行程を理解するために簡単なプログラムを作ってみましょう。

SX-BASICと関連ツールをどこかのドライブ（どこでもいい）に置いて，WIND.Xを起動してください。あとは写真の手順に従って操作するだけです。各部の大きさなどは適当でかまいません。手際よくやれば1分もかからないのではないでしょうか。

プログラムリストを見てわかるように（リスト1），基本的な文法はX-BASICとほとんど同じものになっています。見慣れないのは，頭に“▼”という記号がついている行だけでしょう。

この記号はウィンドウデザイナが生成するウィンドウアイテムの配置とモードの指定に使われるもので，通常のSX-BASICインタプリタの実行時には無視されますので，とりあえず気にする必要はないでしょう（もちろん，よくわからないうちはいじったりしないほうが身のためです）。

慣れが肝心ですから，とりあえず触ってみて，少しずつ攻略していってみてください。

Interface）指向のシステムです。しかし，SX-BASICは基本的にテキストベースのシステムです。INPUT文やPRINT文を用いて，文字列の入出力を行うことはできますが，コントロールマンを呼び出したり，テキストマンを呼び出したりするには少々不便です。

そこで，コントロールを配置する場合にはSX-BASICからタスク間通信を用いて，このウィンドウエンジンに，コントロールを描画する命令を送ります。SX-BASICのウィンドウにコントロールを描画する代わりに，このウィンドウエンジンに描画させるのです。

そうやって，描画されたコントロールに対し，なにかの操作（マウスのクリックなど）が加えられたときには，SX-BASICに対し，タスク間通信を用いてその旨を伝えます。SX-BASICは，そのメッセージを受けてそれに対応する動作を行います。

ウィンドウエンジンとSX-BASICはタスク間通信によりメッセージのやりとりを行っているだけで，プログラムは完全に独立して存在しています。メッセージの送り手がSX-BASICである必要はなく，1993年5月号でも述べたとおり，一定のプロトコルさえ守っていれば，さまざまなプログラムから利用できるようになっています。

＊　＊　＊

以上のように，ウィンドウデザイナ，SX-BASIC，ウィンドウエンジンは密接に結びついてシステムを構成しています（図1）。

インストール

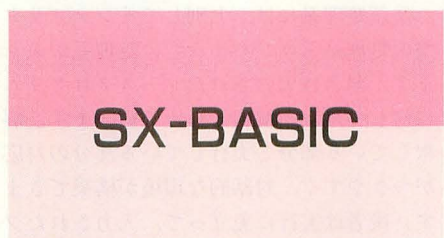
付録ディスクを解凍し，その中から，wind.x，engine.x，sxbasic.x，bedit.xをハードディスク上の適当なディレクトリにコピーしておいてください。

以上でシステムのインストールは完了です。

ハードディスクなしのシステムでもフロッピーディスクだけで動作しますが，あまりおすすめはできません。

プログラミングの実際

準備が整ったところで，いよいよ実際のプログラミングに入りましょう。まず，第1ステップとして，現在の時刻を表示するプログラムを作ってみます。42ページに，一連の作業を行っている様子を，写真つきで紹介しているので，これに沿ってプログラミングを行ってください。

参考までに，この作業で作成される「時計.sxb」の内容はリスト1のようになります。

SX-BASIC

なにぶん，システムが大きいので，今月号だけで全部を説明することはできません。今月号では，システムの中心となるSX-BASICインタプリタに関して説明します。残りの部分に関しては，デバッグ情報とともに来月号以降で掲載していく予定です。

ちなみに，ウィンドウデザイナは，1993年10月号の「秋祭りPRO-68K」に収録されたバージョンとほぼ同じ操作体系を持っています。付属のエディタ（BEDIT.X）も，常識的な操作体系を持ったスクリーンエディタです（セーブは自動的に行われるので，ウィンドウデザイナの状態にかかわらず，いつクローズしてかまいません）ので，とりあえず，次回までノリとほかのSXプログラムからの類推でなんとか使ってください。

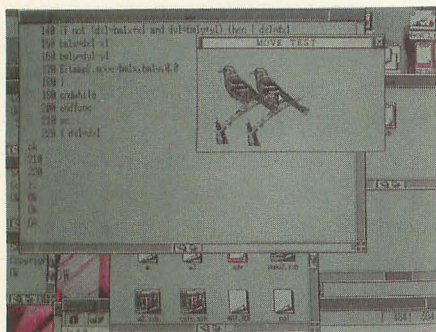
SX-BASICの位置づけ

X-BASICは，screen命令やline命令で画面に対し直接操作を行うことが可能でした。しかし，SX-BASICには画面に対し直接操作を行える命令は，print文だけです。

X-BASICはほかのプログラムと同時に使用されないシングルタスク型のシステムでしたが，SX-BASICはSX-WINDOW上で動作するマルチタスク型のシステムです。当然，SX-BASICと同時に，ほかのプログ

図1

リスト1

```
▼Window Size (240,100),0,0,簡単な時計
/* ここで，初期化に必要な処理を行ってください
while 1
  if Text1.caption <> time$ then Text1.caption = time$
endwhile
func File_Drop(filename;str)
endfunc
▼1,Text1 (24,20,212,92),2,1,3,0,0,1,
```

ビットマップも扱える

ラム（タスク）が実行されている可能性があります。ですから，SX-BASICは勝手に画面モードを切り換えたり，画面を消去することはできないのです。

さらに，SX-BASIC自身のウィンドウは，基本的に文字列表示用のものです。したがって，このウィンドウにボタンやボリュームなどのコントロールを配置することはできません。

しかし，SX-BASICはタスク間通信を用いることによって，ウィンドウエンジンと密接に結びついています。ボタンやボリュームなどのコントロールを使用したい場合にはSX-BASICのウィンドウではなく，ウィンドウエンジンのウィンドウに配置します。また，ウィンドウエンジン上に配置された，これらのコントロールに対して，マウスあるいはキーボードによる操作が加えられた場合には，その情報がすぐさまSX-BASICへ転送されます。このため，ユーザーはSX-BASICの存在を意識することなく，ウィンドウを操作することができます。

SX-BASICの拡張性

前述のように，SX-BASIC自身は，画面に対する入出力機能をほとんど持っていません。外部関数をまったく組み込んでいない状態でのX-BASICも同様です。

しかし，X-BASICの場合は，外部拡張関数により，グラフィックやマウスの制御を可能としています。これに対し，SX-BASICでは，ウィンドウエンジンを拡張することにより，SX-BASICの機能を拡張していく方法を奨励します。

SX-BASICでは，後述のプロパティ設定/参照機能を，SX-BASIC内で解釈することなく，ほとんどそのままの状態でウィンドウエンジンに渡します。引数のエラーチェックなどは，ウィンドウエンジン内で行われます。このため，SX-BASIC本体に改造を加えることなく，機能の拡張を行うことが可能となっています。

中間コードインタプリタ

言語処理系には，大別してインタプリタ型の処理系とコンパイラ型の処理系があります。前者は入力されたソースプログラムに対して逐次，解釈を行っていきます。解釈している部分と実行している部分の対応がつきやすく，対話的な環境が構築できます。後者は実行に先立って，入力されたファイルをすべて翻訳します。そのため，ソースプログラムを解釈する時間がプログラムの実行時間とは別に存在しますので，比較的高速なプログラム実行が可能です。

SX-BASICは入力されたソースプログラムをいったん中間コードへと変換し，その後，中間コードを逐次実行していくという方法を採用しているため，両者の特徴を兼ね備えています。

現在のシステムでの不備

●複数の引数をとるINPUT文

現在のSX-BASICはINPUT文の引数としてひとつの変数しかとれません。

```
input "座標を入力してください(X,Y)";x,y
```

というような命令を現在のバージョンで実行するには，

```
input "X座標を入力してください";x
input "Y座標を入力してください";y
```

のように書き換える必要があります。

●テキストアイテムの編集可プロパティ

現在のシステムでテキストアイテムの編集可プロパティの変更を行うことは避けてください（参照はかまいません）。次回のバージョンまでにはなんとかします。

現在までの経緯とこれから

去年の秋頃のこと，アクセラレータボードの作成が突然いきづまりました（現在も進んでいるとはいえませんが，単にスケジュールの問題です。すみません）。ほんの一瞬，動作するのですが，すぐに「バスエラー」や「不当な命令を実行しました」と出てしまうのです。これはもう，理由は明白でノイズにやられているとしか考えられません。

世の中の常として，このようなときには多層基板を用いるのですが，私もワッセワッセと本を調べて，近所の基板屋さんまで，製作をお願いにいきました。

で，およそ10日ほど，ポッカリと暇な時間ができてしまい，暇潰しにX-BASIC互換のインタプリタを自前で作り始めたところ，意外とスムーズに作成できてしまいました。

驚きとともに，勢いあまって，以前Mook用に作ったSX-BASIC ver.0.1のインタプリタ部分と交換してみたら，そこそこ動いてしまったというのが今回のバージョンです。

ですから，今回のバージョンは，本誌でたびたび名前の出ていたSX-BASICとは，ほとんど別ものです。ver.0.1は編集部内でかなりの期間，動作チェックを受けましたが，今回のバージョンは，締切直前（直後だったかな？　再びすみません）までデバッグを繰り返してきました。プログラムの骨格をいじるような大手術も何回か行いました。

使用に関する注意点（上級者向け）

●SXTASK.Xに関して

現在，このシステムはファイル名の入力などに関して，ダイアログマンを介さずに，自前でウィンドウを作成するダイアログを使用しています。これらのウィンドウに対する親タスクのIDは実際の値よりも0x100大きい値となっています。

現時点では，細かい理由はよくわからないのですが，SXTASK.X（ハンドルネーム：OUZAK氏作成）との共存に関して，あまり相性がよくありません。SX-BASICのシステムがSXTASK.Xよりも下流にある場合，ダイアログへのキー入力ができなくなることがあります。

●プログラムを再コンパイルされる方へ

当プログラムは，GNU C Compilerを用いて作成しましたが，この際GNULIB.Lのバージョンが古いと，変数のキャストを失敗する場合があります。コンパイルの際には，なるべく「GCCによるX680x0ゲームプログラミング」（吉野智興，ソフトバンク刊）の付録ディスクに収録されたもの，あるいはこれよりも新しいバージョンのものを使用してください。

そんなわけで，おそらくバグはまだ大量に潜んでいるものと思われます。

最後に，各プログラム作成に当たって，

gcc ver. 1.00 Tool# Based on 1.42
has version 2.55
hlk version 2.27

を使用しました。

また，ウィンドウデザイナ，ウィンドウエンジンの作成に関して，シャープSX-WINDOW開発キットの初期サンプル版に収録されていた「サンプルドロー」を大幅に参考にさせていただきました。

これらの作者・移植者の方々にこの場を借りて深くお礼申し上げます。

＊　＊　＊

今月は，とりあえずひととおり使えるようにと，駆け足で説明してきました。より込みいったプログラムを作成するには，さらに多くの事柄を説明しなければなりませんし，システムのデバッグを行わなくてはならないでしょう。

次回（来月？）から，項目ごとに説明を行い，適宜簡単なサンプルプログラムを作成したり，デバッグ情報を掲載していく予定です。それでは皆さん，SX-BASICのデバック協力もよろしくお願いします。

SX-BASICリファレンスマニュアル

●データ型

SX-BASICは，以下に示すようにX-BASICと同じデータ型を用意しています。ただし，オーバーフロー，アンダーフローのチェックはしていないので，数値範囲の監視はユーザーがプログラム中で明示的に行ってください。

char	1バイト整数
int	4バイト整数
float	8バイト実数
str	文字列

また，数値の配列変数は3次元まで，文字列の配列変数は2次元まで使用可能です。

●変数の宣言

関数に先立って，プログラムの始めで宣言されるのがグローバル変数，関数内で宣言されるのがローカル変数となります。X-BASICと同様，ローカル変数を使って再帰呼び出しを用いたプログラミングが可能です。

また，X-BASICと違い，宣言されていない変数はまったく使用できません。使用する変数はグローバル変数/ローカル変数にかかわらず，すべて明示的に宣言してください。

SX-BASICの操作方法

●起動オプション

起動オプションは，「/」もしくは「－」の後ろに続けて，以下に示すアルファベット1文字で指定します。

－Ffilename

filenameで指定されたファイルを読み込み，実行します。

－Rn

起動時実行フラグを指定します。このフラグが0のとき，「－F」オプションで指定されたファイルはSX-BASICに読み込まれるだけですが，0以外の数値が指定された場合には読み込みと同時に実行を開始するようにします。

デバッグが終わったプログラムを自動実行させたい場合にはこのオプションを指定してください。

SX-BASICからこのフラグを設定するには，ダイレクトモードで，

autoexec

と入力してください。

なお，このオプションは（ネイティブ）コンパイラ発表後，廃止される可能性があります。

●キーボードショートカット

OPT.1＋Q	SX-BASICの終了
OPT.1＋S	プログラムのセーブ
OPT.1＋L	プログラムのロード

拡張された命令

SX-BASICでは，行番号をプログラムの入力時にしか用いません。そのため，プログラム中の特定の箇所を示すには，ラベルを使用します。

●goto ラベル名

プログラムの制御を強制的に「ラベル名」で示されるところに移します。ただし，関数の外へは飛び出せません。

●ラベルの定義

関数内の行の先頭で，

ラベル名：

とするとラベルが定義できます。ラベル名に使えるのは　最初の1文字が英大小文字で，それ以降が英数字から成り立つ文字列です。

拡張された関数

●alart(式，文字列)

戻り値：int型

式で指定された形式のエラーメッセージ用簡易ダイアログを開き，その内部に与えられた文字列を表示します。アイテムが左クリックされるのを待ち，戻り値としてアイテム番号，あるいはエラーが生じた場合にはリザルトコードを返します。指定するダイアログは，以下のような種類があります。

上位バイト：パターンモード

&h00	黄フラッグ
&h01	赤フラッグ
&h20	F1クラッシュアニメーション
&h80	黒フラッグ

下位バイト：ボタンモード

&h01	確認
&h04	はい/いいえ
&h05	登録/終了
&h06	実行/取り消し
&h07	継続/中止

●getmes()

戻り値：str型

SX-WINDOW上のタスク間通信機能を用いて送られてきたメッセージのうち，SX-BASICの書式にあったものを文字列として返します。具体的には，構造体tseventのwhat2がSX_BASIC_SENDであるメッセージの，whomの値のメモリハンドルで指し示されるデータ列を文字列と見なします（来月号以降で改めて詳しく説明する予定ですが，本誌1993年5月号「タスク通信の可能性」でも触れていたプロトコルです）。

●di()

戻り値：なし

制御がSX-BASICからSX-WINDOWに戻ることを禁止します。ei()命令を実行するまで，ほかのタスクの動作は停止します。

→ei()

●ei()

戻り値：なし

制御がSX-BASICからSX-WINDOWに戻ることを許可します。それ以前にdi()命令が実行されていなければ意味がありません。

→di()

●fock(文字列)

戻り値：int

与えられた文字列をコマンドラインと見なして実行します。この際に起動したタスクのIDを返します。起動できなければ，その理由をダイアログに表示し，戻り値として0を返します。

●TSEnd（式）

式で示されるIDを持つタスクの動作を終了させます。

●sendmes(式，文字列)

式で示されるIDを持つタスクに対し，文字列をSX-BASICの書式に則りメッセージを送信します。

●inputbox$(文字列)

戻り値：str型

与えられた文字列をタイトルに持つ文字列入力用ダイアログを開き，そこに入力された文字列を返します。

●peek(式)

戻り値：int型

式で示されるアドレス内容を返します（バイト単位アクセス）。スーパーバイザ領域にもアクセスできます。

●poke(式1，式2)

戻り値：なし

式1で示されるアドレス内容に式2の値を代入します(バイト単位アクセス)。スーパーバイザ領域にもアクセスできます。

●ref_reg(式)

戻り値：int

戻り値として，式で示されたレジスタの値を返します（章末の式とレジスタの関係参照のこと）。

●set_reg(式1，式2)

式1で示されたレジスタに式2の値を代入します（章末の式とレジスタの関係参照のこと）。

●iocs(式)

式で示された値をmc68000のd0レジスタに代入して，trap #15を実行します。

例）

```
print "ドライブ0をイジェクトします"
set_reg(1,&H9000)
iocs(&H4f)
```

なお，コマンドラインエミュレーションドライバなどを組み込んである場合には，IOCSコールはそのほとんどがマスクされているので，この命令も無効化されてしまいます。

●A_line(式1，[式．サイズ] ……)

戻り値：int

「式」の値を「サイズ」のサイズでmc68000のスタックに積み，式1で示されたmc68000の命令を実行します。パラメータの数は可変数。なお，サイズの指定は以下のとおりです。

b	バイト単位
w	ワード単位（2バイト）
l	ロング単位（4バイト）

例）

```
A_line(0xff23,7.w,0.w);
```

7をワードサイズでスタックに積み

0をワードサイズでスタックに積み

未定義命令ff23を実行する

→標準出力にキャラクタコード7を出力

→ベルを鳴らす

※レジスタの操作は，バッファに対してのみであり，iocs関数，A_line関数実行および終了時にmc68000のレジスタとの整合をとります。SX-BASIC実行中に，レジスタの値を直接，変更/参照するわけではありません。

式とレジスタの関係

式	対応するレジスタ
0	d0
1	d1
2	d2
3	d3
4	d4
5	d5
6	d6
7	d7
8	a0
⋮	

以下同様

拡張されたシステム変数

●mousex

マウスのX座標をローカル座標で返します。

●mousey

マウスのY座標をローカル座標で返します。

●mousel

マウスの左ボタンの状態を返します。押されていれば1を，押されていなければ0を返します。

●mouser

マウスの右ボタンの状態を返します。押されていれば1，押されていなければ0を返します。

プロパティ

SX-BASICは，ウィンドウデザイナ，ウィンドウエンジンと組み合わせることによって，SX-WINDOW上の動作するウィンドウプログラムを記述することができます。このようにして記述されたウィンドウ上のアイテムに対して，命令を行う場合は，アイテムのプロパティを変更することによって行います（例，テキストの色を変える，フォントの大きさを変えるなど）。

SX-BASICから設定できるプロパティには，以下のようなものがあります。

●arrange　　int型

テキストの行揃えモードを設定します。設定できる値は，

-1：右寄せ

0：左寄せ

1：センタリング

の3種類です。

このプロパティを持つアイテムは，テキストだけです。

例）Text1.arrange = 1

●backcolor　　int型

テキストの背景色を設定します。設定できる値は，

0：黒

1：濃いグレー

2：薄いグレー

3：白

4：緑

5：赤

6：黄

7：青

の8種類です。

このプロパティを持つアイテムは，テキストだけです。

例）Text1.backcolor = 2

●caption　　str型

アイテムのキャプションを設定します。長さは90文字までです。91文字目以降は無視されます。

例）Text1.caption = "Hello World !!"

●drag

ファイルアイコンのドラッグ受け入れを行うかどうかを設定します。このプロパティに，0が設定されていると，ドラッグ受け入れは禁止されます。0以外の値が設定されていると，ドラッグ受け入れ処理が行われるようになります。

ファイルアイコンがドラッグされると，プログラム中のWindow_Dropという関数が呼び出されます。この関数は引数にドラッグされたアイコンのファイル名をとります。

ただし，複数のアイコンが一度にドラッグされた場合はアイコンリストの先頭に登録されているアイコンのみを受け付けます。

このプロパティを持つアイテムは，windowだけです。

例）window.growbox = 1

プログラム

```
func Window_Drop(filename; str)
  print filename;"がドラッグされました"
endfunc
```

●enable　　int型

アイテムの使用許可フラグを設定します。0で使用不可，それ以外で使用可となります。オルタネートボタン，チェックボタンに対してこのプロパティを設定した場合には，自動的にアイテムの再描画が行われますので，プログラムで再描画を行う必要はありません。

例）Vol1.enable = 0

●file　　str型

ビットマップの表示するファイル名を与えます。指定できるファイルはパターンエディタ.xあるいはEasydrawSX-68Kで作成されたPAT4形式のファイルのみです。

このプロパティを持つアイテムはビットマップだけです。

例）BitMap1.file = "D:¥FISH.PT4"

●forecolor　　int型

テキストの背景色を設定します。設定できる値は，

0：黒

1：濃いグレー

2：薄いグレー

3：白

4：緑

5：赤

6：黄

7：青

の8種類です。

このプロパティを持つアイテムはテキストだけです。

例）Text1.forecolor = 2

●fontsize　　int型

テキストを表示する際に使用する文字の大きさを設定します。設定できる値は，

0：12ドットフォント

1：16ドットフォント

2：24ドットフォント

の3種類です。

このプロパティを持つアイテムはテキストだけです。

例）Text1.fontsize = 1

●frame　　int型

テキストに枠をつけるかどうか設定します。設定できる値は，

0：つけない

1：つける

の2種類です。

このプロパティを持つアイテムは，テキストだけです。

例）Text1.frame = 1

●growbox　　int型

マウスによる，ウィンドウの拡大/縮小処理を行うかどうかを設定します。このプロパティに0が設定されていると，拡大/縮小処理は禁止されウィンドウの大きさは固定されます。0以外の値が設定されていると，マウスによる拡大/縮小処理が行われるようになります。

このプロパティを持つアイテムはwindowだけです。

例）window.growbox = 1

●height　　int型

テキストとレクタングルとで，このプロパティをそれぞれ違った意味あいで持っています。

テキストの場合，文字の改行幅をドット単位で指定します。

レクタングルの場合には，「彫り」の深さを設定します。正の値でレクタングルが飛び出して見えるように描画され，負の値で彫り込まれているように沈んで描画されます。-5以上5以下の値を設定してください。

このプロパティを持つアイテムはテキストとレクタングルだけです。

例）Rect1.height = -1

●max　　int型

アイテムの持ちうる値の最大値を設定します。minおよびvalue以下の値に設定してください。

ボリュームに対してこのプロパティを設定した場合には自動的にアイテムの再描画が行われますので，プログラムで再描画を行う必要はありません。

例）UpdwnBtn1.max = 3

●menu

該当するアイテム上でマウスの右ボタンがプレスされた際に表示するメニューの内容を設定します。

カンマ「,」で区切られた項目名を並べた文字列を指定します。この際に以下に示す特殊文字が使用可能です。

^

ショートカット文字の指定。次の1文字がショートカット文字となります

_

この文字で始まるメニュー項目はインアクティブ（マウスからの選択不可能）となります

!

項目の頭にチェックマークをつけます

なお，このプロパティに空文字列を指定するとメニューは表示されなくなります。

（注：現バージョンではこのプロパティは設定のみ可能となっています。参照は行えませんので，別に文字列変数に記憶しておいてください。また現バージョンでは，メニュー中で表示したショートカットキーは，メニュー中に表示されるだけとなっています。キー入力による処理の振り分けは，プログラム中で行うようにしてください）。

例）Bitmap1.menu ="ITEM1,^ITEM2,_ITEM3,!ITEM4"

上のように設定されていた場合，プログラム実行時に，ビットマップアイテムBitmap1上で右プレスを行うことにより，図2のようなメニューが表示されます。

●min　　int型

アイテムの持ちうる値の最小値を設定します。valueおよびmax以下の値に設定してください。ボリュームに対してこのプロパティを設定した場合には自動的にアイテムの再描画が行われますので，プログラムで再描画を行う必要はありません。

例）UpdwnBtn1.min = 0

●mode　　int型

ビットイメージアイテムを描画する際の描画モードの指定を行います。描画モードの内訳は以下のとおりです。

0：標準
1：反転
2：強調
3：強調反転
4：消去
5：？？
6：網掛け
7：網掛け反転
8：不可視
9：不可視反転

このプロパティを持つアイテムはビットマップだけです。

例）AlterBtn1.visible = 0

●value　　int型

アイテムの持つ値を設定します。オルタネイトボタンとチェックボタンは，0でOff，それ以外でOnとなります。それ以外のアイテムは後述のmin，max値の範囲内に設定してください。ボリュームに対してこのプロパティを設定した場合には，自動的にアイテムの再描画が行われますので，プログラムで再描画を行う必要はありません。

例）Vol1.value = 20

●visible　　int型

アイテムの可視フラグを設定します。0で不可視，それ以外で可視状態となります。

例）AlterBtn1.visible = 0

これらのプロパティは，すべてのアイテムに対して有効なわけではありません。たとえば，標準ボタンの背景色は濃いグレーに固定ですし，文字サイズは12ドットに固定されています。ですから，標準ボタンに対して，これらのプロパティを設定しようとしても，エラーになるわけです。

どのようなアイテムが，どのようなプロパティを持っているのかを以下に示します。

id	アイテムの種類	デフォルトの設定
1	テキスト	forecolor backcolor fontsize height editable arrange visible
2	レクタングル	height visible
3	標準ボタン	visible
4	ボリューム	min max value visible
5	オルタネートボタン	value visible
6	チェックボタン	value visible
7	アップダウンボタン	min max value editable visible
8	ビットマップ	mode file visible

メソッド

背景色や文字の大きさなどを変更するときには，SX-BASIC上からプロパティを設定してやるのは前述のとおりです。

プロパティを設定するということは，アイテムに対してほんの少し変更を加えるといった意味あいでした。これに対し，メソッドというのはアイテムに対し働きかけるときに使う手段です。

SX-BASICには，以下の種類のメソッドがあります。

●delete

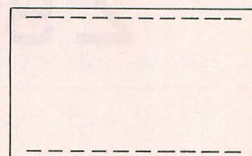
アイテムの消去を命令します（引数，戻り値，共になし）。

例）AlterBtn1.delete

AlterBtn1というアイテムを消去します。

●move = x1,y1,x2,y2

（x1,y1,x2,y2はともにint型）

アイテムの表示位置を指定します。アイテムは，x1,y1,x2,y2で示されるレクタングル中に表示されます。

例）StnBtn1.move = 10,10,10＊40,10＊40＋10

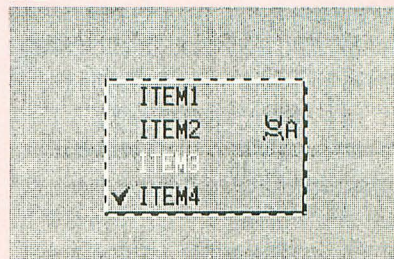
●new = id, x1,y1,x2,y2

（id,x1,y1,x2,y2はともにint型）

x1,y1,x2,y2で示されるレクタングルに，アイテムを新たに作成します。idにより，作成するアイテムの種類を指定します。その内訳は以下のとおりです。

id	アイテムの種類	デフォルトの設定
1	テキスト	forecolor=3 backcolor=1 fontsize=1 height=17 editable=0 arrange=0 visible=0
2	レクタングル	height=0 visible=0
3	標準ボタン	visible=0
4	ボリューム	min=0 max=100 value=50 visible=0
5	オルタネートボタン	value=0 visible=0
6	チェックボタン	value=0 visible=0
7	アップダウンボタン	min=0 max=0 value=0 editable=0 visible=0
8	ビットマップ	mode=0 file="" visible=0

newメソッドにより作成されたアイテムは，visibleプロパティが0となっており，そのままでは画面に表示されません。各種プロパティ設定を行ったあと，visibleプロパティを1に設定することにより，画面に表示させてください。

図2

EX-WINDOWによる拡張ツール

Z's-EX&MATIER-EX

Kikuchi Isao 菊地 功

お馴染みのグラフィックツール拡張キットZ's-EXの新バージョンの発表です。同時にMATIERにも対応したMATIER-EXも発表します。操作方法は同じで共通の外部ファイルが使用できます。

はじめに

X68000は発表されたときからハイカラー(65536色)の表示という，当時としては驚異的なスペックを誇っていました。それゆえ，そのハイカラーを利用した多くのグラフィックツールが発表されました。なかでも初期のうちから発売され，多機能ゆえに多くのユーザーに利用されたZ's STAFF PRO-68Kというグラフィックツールがあります。これは現在でもver.3となって，広く使用されています。

また，かなり遅れてMATIERというグラフィックツールも発表されました。こちらも多少Z's STAFFとは性質が異なるものの，かなり広く利用され，現在ではver.2がリリースされています。

この2つが現在X68000で利用されている市販グラフィックツールの双璧といってもいいでしょう。Z's-EXとMATIER-EXは，それらのグラフィックツールを拡張するためのツールなのです。

Oh!Xを以前から読んでおられる方はZ's-EXはご存じでしょう。元は丹明彦氏によってOh!X1991年1月号の付録に掲載されたものです。その後ver.1.1により外部ファイルを使用できるようになりましたが，このバージョンは応急的に外部ファイルを扱えるようにしただけで，Z's-EXとのインタフェイスはきわめて貧弱なものだったのです。

今回のZ's-EX ver.2では，これらの問題を解決するために，Z's-EX側に割り込みファンクション(EXコールとでも呼びましょう)を持たせ，外部ファイルから手軽にZ's-EX内部の機能を利用できるようにし，Cのライブラリも添付しました。これにより，外部ファイルの作成が手軽にできるようになりました。

MATIER-EXは今回が初めてですが，ご想像のとおりZ's-EXをMATIERに移植した(?)ものです。今回は2つのバージョンを配布しましたが，使用するMATIERのバージョンと同じものを使用してください。どちらのバージョンも機能は同じですが，MATIER側の相違を吸収するための策です。

Z's-EXもMATIER-EXも機能はまったく同じですし，外部ファイルを作る際もほとんど意識する必要はありません(EXコールを使用する限り)。そこで，Z's-EXとMATIER-EX(と将来ほかのグラフィックツールに移植されるかもしれないものも)をEX-WINDOW(えっくすうぃんどう)と呼ぶことにしましょう(はは……，怒られるかなぁ)。

改良点

Z's-EX Ver.1.1からの変更点としては以下のものがあります。

外部ファイルを階層状に登録できるようになった

大量の外部ファイルを登録するときなどに，分野別に階層を分割しておけば，目的の外部ファイルがすぐに見つけられるでしょう。ディレクトリのようなものです。

外部ファイルからEX-WINDOWの内部関数を呼べるようになった

前述のEXコールです。TRAP7を使用しています。

マスクを8ビットに拡張した

実際に使用できるのはそのうち5ビットですが，マスクの濃度によって透過率が異なります。従来のマスクは不透明マスク(マスクコード0)として扱われます。以後，このマスク機能はアナログマスクと呼ぶことにします。

裏画面およびアナログマスクを切り放せるようになった

ユーザーメモリが少ないときには，アナログマスク・裏画面の順で切り放します。切り放した場合は当然それらの機能は使えません。また，起動オプションによっても切り放し可能です。

いくつかの割り込みをトラップした(Z's-EX)

Z's STAFFでバスエラーやアドレスエラーを起こすと，「未登録の割り込みです」とメッセージが表示され，復帰できなくなりますが，Z's-EX側でトラップすることにより，復帰できるようになりました。その後の動作は保証できませんが，外部ファイルの開発段階においてはありがたく感じることでしょう。

使用環境およびインストール

Z's-EXとMATIER-EXでは，若干使用環境が異なります。

●Z's-EX

基本的にどのバージョンのZ's STAFFでも動作しますが，ver.1では動作が不安定になる場合があるようですので，ver.2以降で使用することをおすすめします。

インストール方法ですが，Zs_EX.XとZs_EX.SYSはZ's STAFFの本体であるSTAFF.Xと同じディレクトリに入れておいてください(Zs_EX.SYSについてはあとで説明します)。外部ファイルはカレントディレクトリか，もしくはパスが通っていればどこに置いてあってもかまいません。

インストールができたら起動ですが，必ずZs_EX.Xがあるディレクトリにカレントを移してからZs_EX.Xを起動してください(STAFF.Xをユーザーが起動する必要はありません)。Zs_EXのオプションを

以下に示します。

/t

裏画面とアナログマスクを切り放すタイニーモデルを起動します。

/s

アナログマスクを切り放すスモールモデルを起動します。

/f<filename>

外部ファイルを登録するコンフィグファイルを指定します。デフォルトはZs_EX.SYSです。

/x<filename>

ショートカットキー(CTRL+起動キー)で起動する外部ファイルを指定します。デフォルトはPICFILERです。

/n

起動キーによる起動を禁止します。Z's STAFF ver.3から呼ぶ外部ファイルがIOCSコールB_BITSNSをしている場合に指定してください。このオプションを指定した場合は, Zs_EXはZ's STAFF ver.3の外部ファイル実行でEXOPEN.Xを実行して起動することになります。

上記以外のオプションを指定すると, オプションの一覧を表示します。

また, 正常に起動できない場合は, 以下のようなことが考えられます。

・メッセージが表示され, 終了する。

メッセージのとおりです。

・Zs_EXが起動されたあと, 黙ってしまう。

Zs_EX.SYS (後述) の記述が間違っていると思われます。

・真っ白になって, 飛んでしまう。

Z's STAFFがメモリ確保に失敗したと思われます (きわどいとハングするようです)。小さなモデルを起動するか, Zs_EX.SYSで外部ファイル実行用メモリを減らす, もしくは思い切って大きくしてみてください。

上記以外にも常駐物との相性なども考えられます。また, Zs_EXが正常に動作したように見えても, 終了後に常駐物が動作しなくなる, 常駐解除できなくなることなどもありえます (特にキー割り込みを使用している常駐物) ので, できる限り常駐物はZs_EX起動前に常駐解除しておくことをおすすめします。

さて, 無事起動したかと思いますが, 一見したところなんの変哲もないZ's STAFFが起動しただけです。ここでおもむろにSキーを押してみましょう。怪しげなウィンドウが現れましたね。これがZ's-EXの基本となるルートウィンドウです。ただし, このウィンドウを表示させる前には, 必ずZ's STAFFのウィンドウはすべて閉じておいてください (タイトルバーは表示させたままでかまいません)。また, 起動キーはコンフィグファイルを書き換えることによって, ほかにキーに割り当てることもできます。

以後の操作はMATIER-EXと共通ですので, またあとで説明することにしましょう。

●MATIER-EX

前述のとおり, MATIERのバージョンによって使用するMATIER-EXのバージョンが異なります。MATIER ver.1とver.2にはそれぞれ同じバージョンのMATIER-EXを使用してください。

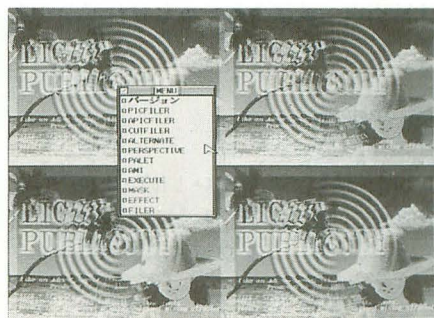
インストールについては, 本体のMAT_EX.XおよびMAT_EX.SYSはMATIERの本体であるMAT.Xと同じディレクトリに入れてください。また, 外部ファイルですが, 必ずパスの通ったディレクトリに入れておいてください。これは, MATIERのファイラーがカレントを移動してしまうためです (Z's-EXも以前はそうでしたが, 「STAFF.TMPがあちこちにできて困る」という苦情があったため, カレントを移動しないようになりました)。

起動に関しては, パスが通っていればMAT_EX.Xのあるディレクトリに移動しなくてもかまいませんが, 必ず環境変数MATIER(必ず大文字で)にMAT.Xのあるディレクトリを設定しておいてからMAT_EX.Xを実行してください。おっとその前にMATIERのver.1を使用されている方は, MAT.BUFというMATIERの環境設定ファイルの中で子プロセス用のメモリを300~500Kバイト程度確保しておいてください (MAT.BUFの記述についてはMATIERのマニュアルに譲ります)。それではMAT_EX.Xのオプションを以下に示します。

/s

アナログマスクを切り放すスモールモデルを起動します。

/f<filename>

外部ファイルを登録するコンフィグファイルを指定します。デフォルトはMAT_EX.SYSです。

これがルートウィンドウ

裏画面はMATIERの裏画面と共通ですので, タイニーモデルはありません。では, MATIERを起動したあと, EX-WINDOWを起動してみましょう。

Z's-EXと違い, MATIER-EXはSキーを押しても起動しません。それではどうやって起動するのでしょう。カスタマイズしていなければF1キーでコマンドモードになるはずですので, そこでEXOPEN<リターン>と入力してみてください。どうです? ウィンドウが表示されましたね。このようにMATIER-EXはEXOPEN.Xという専用のコマンドを使用して起動されます。

しかし, 起動するのにいちいちコマンドモードからキー入力するというのは, あまり美しくありません。そこで, MAT.BUFのファンクションキーコマンドの部分にEXOPENを登録してしまいましょう。これでファンクションキー一発で起動するようになったわけです。一瞬ファンクションキーが表示されますが, 気にしないことにしましょう。

ところで, このEXOPEN.Xは引数をとることができます。MATIERとMATIER-EXが裏画面を共用するのは先ほど述べましたが, MATIERは裏画面を最大4面持つことができます。これに対して, EX-WINDOWは裏画面を1面しか持つことができません。そこで, EXOPENの実行時に引数として1~4の数値を渡すことによって, MATIERの4面の裏画面のうち何面目をMATIER-EXの裏画面とするかを指定できるようになっています。たとえば,

EXOPEN 2<リターン>

とすれば, MATIERの2面目の裏画面をMATIER-EXの裏画面として使用できるようになります (デフォルトは1です)。ただし, その時点で指定された裏画面が

MATIERで確保されていなければなりません。

使用法

さて，それではいよいよEX-WINDOWを使ってみましょう。おそらく画面には図1のようなウィンドウが表示されていることと思います。

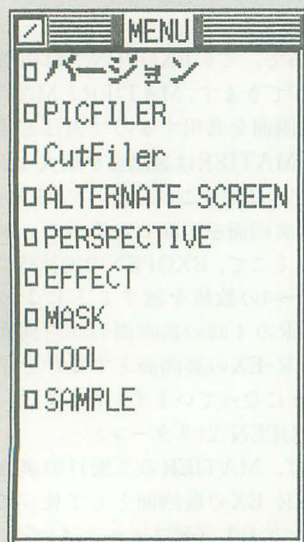
・タイトルバー

ここをつかんでドラッグすることで，ウィンドウを移動させることができます。ウィンドウは画面からはみ出すことはできません。また，真ん中あたりには階層の名前が書いてあります。ルートの場合はMENUとなります。

・クローズボックス

このボックスをクリックすることで，1つ上の階層に移ります。ルートにいる場合にはEX-WINDOWを抜けて，Z's STAFFまたはMATIERに戻ります。また，次のEFFECTウィンドウ内でマウスの左右のボタンを同時に押すことでも同等に機能します。

・EFFECTウィンドウ

コンフィグファイルに登録されているEFFECTおよび階層がずらりと並んでいます。EFFECTは黒で，階層はグレーで表示されていて，それぞれの左端にある四角いボタンをクリックすることで，EFFECTは実行，階層はその階層に移ることができます。

また，EFFECTによっては，右端に数字の入った四角い枠を持つものがあります。これはEFFECTに渡すパラメータで，枠の中でマウスを左右クリックすることで値を増減させることができます。パラメータの意味については，それぞれの外部ファイルの説明を参照してください。

EFFECTや階層の数が多くて，ウィンドウに入りきらない場合は，ウィンドウの真ん中あたりでマウスのボタンを押してみてください。左ボタンでスクロールアップ，右ボタンでスクロールダウンします。

・スペースキー

スペースキーを押すことで，表画面と裏画面が入れ替わります。

・ESCキー

ESCキーを押すことで，いきなりEX-WINDOWを抜けます。次にEX-WINDOWを起動したときは，ESCキーを押したときにいた階層が開きます。

・HELPキー

HELPキーを押すと，テキストビュア(EXTYPE.X)が開いて，ログを表示します(ファイル名はそれぞれZs_EX.LOG, MAT_EX.LOG)。このログは標準出力および標準エラー出力をファイルに落とすものです。Z's STAFFはテキストVRAMを退避画面に使用していますので，不用意にコンソールに出力されると画面が壊れます。Z's STAFFが単独で走っていたのなら，自分が気をつければ済むことですが，ver.3になって外部ファイルをサポートするようになってもなおこの方式は変わっていません。

そこで，EX-WINDOWは（MATIER-EXはZ's-EXとの互換のため）標準出力と標準エラー出力をトラップしてあるのです(リダイレクトと同じようなこと)。しかし，なんらかのメッセージを発する外部ファイルを実行したときに，そのメッセージが見られないと困ることも考えられますので，このような方式にしました。

これがEX-WINDOWの操作の基本です。そうそう，いい忘れていましたが，EX-WINDOWはシングルウィンドウです。期待されていた方，ごめんなさい。

コンフィグファイル

EX-WINDOW自体はあくまでも外部ファイルを起動するためのプラットフォームを提供するだけであり，外部ファイルがなければなんの役にも立ちません。外部ファイルは今回配布したもののほかに，各自で作成することもできますし，一般に出回っているフィルタなどのなかにも使えるものがあるでしょう。今回は外部ファイルの作り方はあと回しにして，新しく外部ファイルを登録する方法について説明します。

外部ファイルを登録するには，コンフィグファイルを読み込ませてやる必要があります。コンフィグファイルはEX-WINDOWが起動する際にメモリに展開され，動作中に再登録することはできません。コンフィグファイルのデフォルト名はZ's-EXの場合はZs_EX.SYS, MATIER-EXの場合はMAT_EX.SYSですが，/fオプションにより変更することができます。

図1　起動直後のメニュー

図2　コンフィグファイルの書き方

```
mem, key
:title1            ←EFFECTの場合
        flag, pn [: p1min-p1max, p1default [: p2min-p2max, p2default]]
        filename1 [option1]
        [filename2 [option2]]
:title2
        flag, pn [: p1min-p1max, p1default [: p2min-p2max, p2default]]
        filename1 [option1]
        [filename2 [option2]]
:title3            ←階層の場合
{
    :title3
            flag, pn [: p1min-p1max, p1default [: p2min-p2max, p2default]]
            filename1 [option1]
            [filename2 [option2]]
}
```

コンフィグファイルはテキストファイルですので，エディタなどで自由に編集することができます。ただし，フォーマットを誤って記述すると，暴走することがありますので，気をつけてください。図2にコンフィグファイルのフォーマットを示します。

mem

外部ファイル実行用メモリのバイト数を10進で記述します。だいたい300～500程度でいいでしょう。Z's-EXでのみ有効ですが，MATIER-EXで指定しても無視されるだけです。

key

Zs_EXの起動キーを2桁の数字で記述します。1桁目がキーコード，2桁目がキーデータを表し，省略時は37，すなわちSキーです。これもZ's-EXでのみ有効で，MATIER-EXでは無視されます。

title

機能名です。EX-WINDOWから外部ファイルを実行する際には，この機能名を選択することになります。文字数は基本的に半角で16文字ですが，パラメータの数によりそれ以下にしなければ表示が衝突してしまいます（機能に支障はありません）。

flag

外部ファイルを起動する前にZs_EXが行う処理を指定します。いまのところ，0でなにもせずに，1で矩形指定を行い座標をパラメータとして外部ファイルに渡して起動します。

pn

Zs_EXから指定できるパラメータの数で，0～2の範囲で指定します。パラメータは1桁の数字で，外部ファイルにもそのまま渡されます。

p1min,p1max,p1default
p2min,p2max,p2default

それぞれ1つ目，2つ目のパラメータの最小値，最大値，初期値です。最大値は最小値より大きくなければならず，初期値はその範囲内でなければなりません。すべて0～9までの1桁の数字で記述します。

filename1
filename2

実際の外部ファイル名を指定します。拡張子は指定しないでください（手抜きのため不都合が生じる場合があります）。ひとつの機能で2つまでの外部ファイルを連続して実行させることができます。パラメータなどはどちらの外部ファイルにも渡されます。

option

外部ファイルごとに固定のパラメータを指定します。パラメータは半角で5文字までです。

インデントはしなくても構いませんが，図以外のところで改行したり，省略しないでください。たとえば'{'や'}'は1行に単独で記述してください。エラーチェックが甘いので，暴走してしまう可能性があります(申しわけありません，手抜きです)。ディスクに付属のZs_EX.SYSおよびMAT_EX.SYSを参考にするといいでしょう。

外部ファイル数に特に制限はありませんが，実際問題として外部ファイル1個につき100バイトのメモリを消費します(階層も同様です）ので，メモリが確保できるだけということになります。

また，当然のことですが，矩形指定やパラメータなどは対応した外部ファイルに対してしか意味がありません。全然考慮していない外部ファイルに対して矩形指定したからといって，その領域だけ処理されるようになることはありません（矩形指定の場合はその領域だけ待避画面に書き戻しますので，結果的にそうならないこともないんですが)。ただ，これらの指定はEX-WINDOWがEXコールを持たない頃にとりあえずでつけたものですので，最近では全然使ってません。外部ファイルの中でパラメータやら領域指定したほうが簡単とはいいませんが，自然ですからね。

アナログマスク

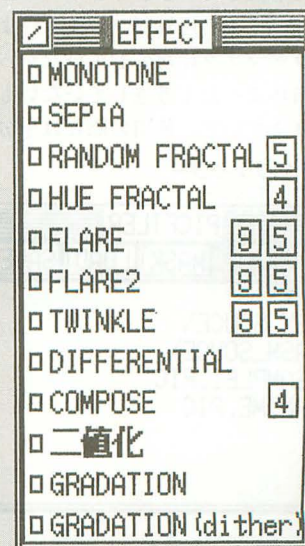
Z's STAFFのマスクは画面では青い点滅で示されています。この点滅はいちいちVRAMを書き換えているわけではなく，マスク部分をカラーコード1で塗っておいて，パレット機能によって点滅させています。しかし，65536色モードでパレット機能を使用した場合，1色だけではなく，ほかの色も引きずられてパレット機能の影響を受けてしまいます。Z's STAFFのように，カラーコード1のパレットを変化させた場合，下位8ビットが01_Hのピクセルもすべて変化してしまうのです。

Z's STAFFでは最下位ビットが立っている(奇数)カラーコードは選択できませんが，マスク以外にもうひとつこの点滅を目にすることができます。そう，グリッドです。グリッドはカラーコード$F801_H$を使うことによりマスクと「同時」に点滅させているのです。Zs_EXのアナログマスク機能はここに目をつけ，マスクの透過率により点滅の色を5ビット32段階に変化するようにしました。8ビット256段階にすることもできましたが，そこまでしてもあまり意味がないでしょうし（RGB自体がそれぞれ32段階ですので)，5ビットのほうが扱いやすいと思ったからです。このアナログマスクは青い従来のマスクは不可視で，緑に近くなるほど透過率が高くなるようにしています。

しかし，Z's STAFFにはもちろんこの機能はありませんので，アナログマスクが表示されている状態でZ's STAFFに戻ると，すべて不可視マスクの表示に戻ってしまいます。アナログマスク情報は別の領域に保存してありますので，表示がおかしくなっても誤動作することはありませんが，問題となるのはアナログマスクがあるはずの領域のマスクをいじられたときです。Zs_EXからはZ's STAFFでどのようなマスク操作を行ったか知る術がありません。

このまま作業を続ければマスクに混乱をきたします（飛んだりすることはありませんが)。そこで，Z's STAFFでマスクを操作したあとにZs_EXを起動する場合は，OPT.1キーを押しながら起動キーを押して

図3 パラメータの例

EFFECT
MONOTONE
SEPIA
RANDOM FRACTAL 5
HUE FRACTAL 4
FLARE 9 5
FLARE2 9 5
TWINKLE 9 5
DIFFERENTIAL
COMPOSE 4
二値化
GRADATION
GRADATION(dither)

ください。また，マスク操作はできる限り付属の外部ファイルで行うようにしてください。

で，MATIER-EXですが，MATIERはマスク表示の方式がZ's STAFFと違うばかりか，ver.1と2でも異なっています。それをいちいち考慮して外部ファイルを作っていたのではたまりません。そこで，MATIER-EXはウィンドウを起動するときにマスク表示をZ's STAFF互換に書き直すことにしました。マスクの下が見えないと，不便なこともあるかもしれませんが，これはZ's-EXとの互換だけでなく，アナログマスクを使用するためでもあるのです(MATIERの方式ではマスクですべてのビットを使用するので)。起動時に若干間が空きますが，これも気にしないことにしましょう。

注意点は，Z's-EXとほぼ同じですが，MATIER-EXにはOPT.1キーを押しながら起動する機能をつけ忘れたことをいま思い出しました，ははは……。ごめんなさい。次の機会まではつけときます，ということで，MATIER-EXでアナログマスクを使用するときには注意してください。

外部ファイルについて

さて，EX-WINDOWの使い方がわかったところで，今度は付属の外部ファイルについて説明しましょう。

ではディスクに収録された外部ファイルについて，サンプルのコンフィグファイルに従って解説していきましょう（カッコ内が外部ファイル名)。外部ファイルによってはアナログマスクに対応していなかったり，マスク自体まったく考慮していないものなどもありますので，使用の際には十分に注意してください。また，処理に時間がかかるコマンドはたいてい右クリックで中断します(たぶん)。今回収録した外部ファイルのうち，いくつかは過去にOh!X誌上で説明がなされたものを改良したものです。そちらも参照してください。

多くのソフトウェアでのマウスの操作法と同様に，たいていは左クリックが「確定」または「実行」，右クリックが「キャンセル」に対応していますが，場合によっては左クリックと右クリックが同じ働きをすることがあります。

ルートメニュー

●バージョン(VERNO.X)

EX-WINDOWのバージョンが表示されます。今回ディスクに収録されたZ's-EXでは「Z's EX ver 2.0」，MATIER-EXでは「MATIER EX ver 1.0」および「ver 2.0」と表示されるハズです。

●PICFILER(PICFILER.X)

X68000ではほぼ標準フォーマットとなった柳沢明氏によるPICフォーマットのローダー/セーバーです（65536色のみ)。また，マスクのロード/セーブも行えます(アナログマスク対応)。

実行すると，図4のようなファイルウィンドウが現れます。クローズボックスやタイトルバーはもう説明の必要はないでしょう。操作法としてはまず，クローズボックスの下の枠をクリックして枠の中のアルファベットをドライブ名に合わせます。すると，ウィンドウ中央の大きな枠の中に拡張子がPICのファイル名とディレクトリ名が表示されますので，意中のファイルがあればそのファイルの左にある四角いボタンをクリック，なければディレクトリに入るなどしてファイルを探してください。

親ディレクトリは'..'，もしくは左右同時クリックでも抜けられますし，Zs_EXのウィンドウと同様に左右クリックでスクロールできます。なにはともあれ，ファイルを指定するといちばん下の細長い枠にファイル名が表示されます。ここで，右上の'LOAD'や'SAVE'ボタンをクリックすることによって，ファイルの操作をします。

しかし，既存のファイル名でしかセーブできないなんて馬鹿なことはありません。細長い枠をクリックして，直接ファイル名を指定することもできます。ただし，ドライブ名やディレクトリ名は指定することができません。また，ファイル名指定時にボタンをダブルクリックすることによってもロードすることができます。

マスクについても同様です。中央上にある'MASK'ボタンをクリックして'PIC'に変わったことを確認してください。これでマスクモードになりました。あとはPICと同様に操作してください。また，▲印のアイコンでフロッピーディスクをイジェクトします（ハードディスクはルートディレクトリに降ります)。

このコマンド実行後はPICFILER.Xと同じディレクトリにPICFILER.EXというレジュームファイルを生成します。環境が変わった場合などはこのレジュームファイルのせいで動作不良になる場合があるようですので，ファイルが表示されない場合などはこのレジュームファイルを削除してください。

●CUTFILER(CUTFILER.X)

月刊電脳倶楽部でお馴染みのカットファイルのローダー/セーバーです。カットファイルはモノクロの二値データですので，内部から後述のMONOTONE.XとTO2.Xを呼び出します。これらの外部ファイルは同じディレクトリに収めておいてください。矩形指定をするほかはPICFRILERと使い方は同じです（矩形指定は内部で行いますので，コンフィグファイルでの矩形指定フラグは立てる必要はありません)。

●ALTERNATE SCREEN(ALTERNATE.X)

裏画面と表画面を入れ換えます（アナログマスクも入れ替わります)。EX-WINDOWのウィンドウが開いている状態で，スペースキーを押した場合と同じ動作にな

図4 PICファイラー

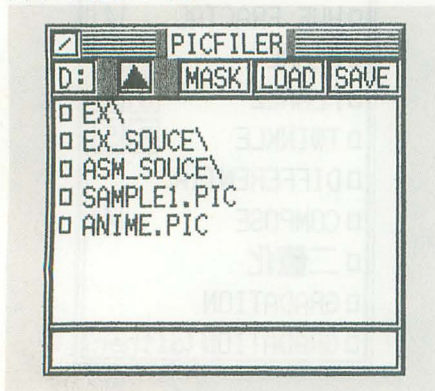
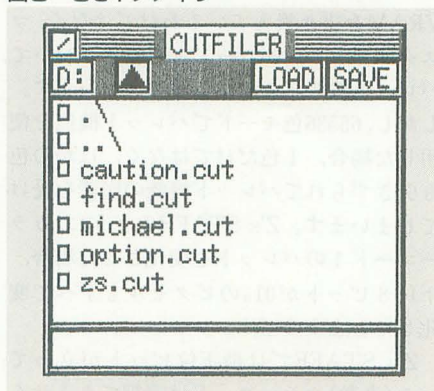



図5 CUTファイラー

ります。

●PERSPECTIVE(PERSPECTIVE.X)

裏画面を1枚の板とみなし，任意の角度などを与えて3次元状に表画面に作画します。実行すると図6のようなウィンドウと裏画面を示す枠が表示されますので，適当にいじってみてください。説明するよりそのほうが早いでしょう。アイコンは左クリックで正動作，右クリックで逆動作します。

また，'RESET'したのちに実行した場合のみ，専用の2D転送により裏画面を表画面にそのままコピーします。詳しくはOh!X 1991年1，2月号を参照してください。マスクについては，裏画面のマスクはすべて不可視マスクとみなしますが（穴が空いたようになる），表画面はアナログマスクに対応しています（処理中は無視しているように見えます）。このコマンドもPERSPECTIVE.EXというレジュームファイルを生成します。

EFFECT階層

●TWIRL(TWIRL.X)

●うにょうにょ(TWIRL2.X)

●収差(SHUSA.X)

中野修一氏による特殊効果フィルタです。

TWIRLは渦巻き状に画面を変形させます。TWIRL2はその亜流で予想不可能な変形を行います。SHUSAは収差変形を行います（Oh!X1993年3月号参照）。

いずれも実行は全画面に対して行われマスクは無視されます。

●MONOTONE(MONOTONE.X)

画像をモノクロ化します。コンフィグファイルでflagを1にして矩形指定を行うことにより，指定した矩形内のみを処理します。矩形指定を行わない場合は全画面(0,0-511,511)が対象になります。オプション/Gはつけてもつけなくてもかまいません。

人間の目にはRGBが同等の明るさには映らないため，モノクロ化するにあたっては，それぞれ適当なウェイトをつけています。詳しくはOh!X1991年1，2月号を参照してください。

●SEPIA(MONOTONE.X)

画像をセピア調にします。オプション/SをつけるGy以外はMONOTONEと同じです。

●RANDOM FRACTAL(FRACTAL.X)

フラクタルによってピクセルを上下にずらし，画像を崩します。処理範囲の指定はMONOTONEと同様です。パラメータはひとつで，0～9の範囲で与えます。値が小さいほど崩れ方が少なく，大きくなるに従って崩れ方が大きくなります。また，この外部ファイルを実行するには同じディレクトリにRF.DATが必要です。詳しくはOh!X 1991年1，2月号を参照してください。

●HUE FRACTAL(FRACTALH.X)

ランダムフラクタルがピクセルをずらすのに対して，色相フラクタルは色相をずらします。それ以外はランダムフラクタルと同様です。詳しくはOh!X 1992年7月号を参照してください。

●FLARE(FLARE.X)

マスク（アナログマスク）の下の色を周辺に滲ませることによって，マスクされた部分にぼんやり光っているような質感を与えます。処理範囲の指定はMONOTONEと同様です。パラメータは2つで，ひとつ目は強さを0～9の範囲で指定し，2つ目は広さを1～9の範囲で指定します。詳しくはOh!X 1991年1，2月号を参照してください。

書き込み時のみアナログマスクを考慮します。

●FLARE2(FLARE2.X)

FLAREは滲ませる輝度の判定範囲を矩形として処理していますので，場合によっては不自然になります。しかし，FLARE2は円で処理していますので，比較的自然な結果が得られます。しかし，当然の如く，FLAREより処理に時間がかかりますので，状況に応じて使い分けるようにするとよいでしょう。使用法はFLAREと同様です。

●TWINKLE(TWINKLE.X)

任意のカラーコードのピクセルに，輝いているような処理を施します。矩形指定により処理範囲を指定しますが，自前で矩形指定を行いますので，コンフィグファイルでflagを立てる必要はありません。矩形を指定したあと，任意のピクセルをクリックすることにより（矩形外でも構いません），矩形内の同色のピクセルに対して処理が施されます。パラメータはFLAREと同様です。

●DIFFERENTIAL(MONOTONE.X,DIFFER.X)

画像に微分処理を施し，レリーフのような質感を与えます。微分処理の本体はDIFFER.Xですが，その前にMONOTONE.Xでモノクロ化しておく必要がありますので，2つの外部ファイルを連続実行させます。処理範囲の指定はMONOTONEと同様です。詳しくはOh!X1991年1，2月号を参照してください。

●COMPOSE(COMPOSE.X)

表画面に任意の比率で裏画面を合成します。処理範囲の指定はMONOTONEと同様です。パラメータはひとつで，裏画面の比率を1～8の範囲で指定します。表のアナログマスクには対応していますが，裏のアナログマスクはすべて不可視マスクとして扱われます。詳しくはOh!X 1992年2月号を参照してください。

●二値化(MONOTONE.X,TO2.X)

桒野式ディザにより，モノクロ二値化します。処理範囲はMONOTONEと同様です。

●GRADATION(GRAD.X)

矩形内をその四隅の色を補間する色でグラデーションフィルします。処理範囲の指定はMONOTONEと同様です。また，'/D'オプションをつけることで，ディザをかけて補間します。詳しくはOh!X 1992年2月号を参照してください。

●CURVE GRADATION(CGRAD.X)

湾曲したグラデーションを描きます。矩形指定をすると（内部で行いますのでコンフィグファイルで指定する必要はありません），その矩形が田の字(？)になり図7のよ

図6 PERSPECTIVE

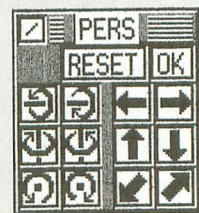
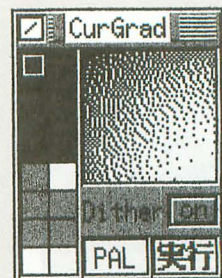
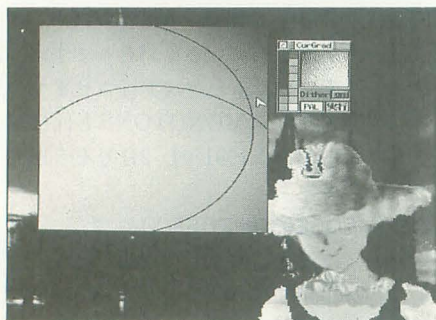



図7 円形グラデーション

円形のグラデーションはこんな感じ

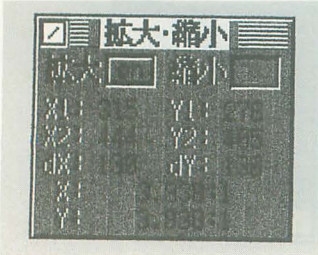
うなウィンドウが表示されます。ウィンドウ外をポイントすると矩形内の十字が湾曲します。この曲線がグラデーションの湾曲の度合いになります。適当に設定したら，パレットから色を指定し，外側の色なら右の四角の左上の隅，内側なら右下をクリックしてください。この枠で色を確認したあと，ディザをかけるかどうかを指定し，実行します。

また，パレットに指定したい色がない場合は'PAL'アイコンをクリックし，PALETウィンドウを呼び出してください。PALETウィンドウの使い方は後述のパレットの項を参照してください（PALET.Xを呼び出しますので，必ずパスを通しておいてください）。

この外部ファイルはアナログマスクに対応しています。

●拡大・縮小(ZOOMIO.X)

適当に補間しながら任意倍率の拡大・縮小を行います。図8のようなウィンドウが開きますので，拡大か縮小を選択したあとに矩形を指定してください。拡大の場合はその矩形範囲を画面全体に拡大し，縮小の場合は画面全体を矩形範囲内に縮小します。ウィンドウ内に表示される数値を目安にしてください。また，読み出す段階ではマスクは一切無視しますが，書き込みではアナログマスクに対応しています。

なお，縦横比は保存されます。

MASK階層

●MASK(MASK.X)

いくつかのマスク操作をひとつのウィンドウにまとめたものです（矩形指定はできません)。実行すると図9のようなウィンドウが開きます。

VIEW

マスクはそのまま，マスクの下を見ます。

REV

マスクを反転します。不可視マスク部分はマスクが剝がされ，マスクのない部分は不可視マスクになります。また，アナログマスクも透過率が反転します。

ALTERNATE

表と裏のマスクを交換します。

OFF

マスクを剝がします。

●MASK PAINT(MASKPAINT.X)

表示画面の色境界を基準としてマスクでペイントします。アナログマスクが使用不可な環境では領域をマウスで指定するだけですが，アナログマスクが使える場合には図10のようなウィンドウが開きますのでレベルメータで任意のマスク濃度を指定したあと，画面上の領域を指定してください。値が大きいほど透過率が高くなります。

また，'OFF'を指定することでマスクを剝がすこともできます。指定中は黄色い点滅で表されますが，右クリックで指定を終了すると，マスク下の画像が表示されます。

●緑→マスク(GTOMASK.X)

作業画面の緑成分をマスクに変換します。緑成分が多いほど，透過率の低いマスク，つまりマスクの値は小さくなります。ただし，そのまま変換するとすべてマスクになってしまいますので（マスクだけで32段階あるから)，緑成分が0のときだけはマスク値31(いちばん透過率の高いマスク)にせずに，マスクをしないようにしています。また，アナログマスクが使用できない環境では緑成分0以外は不可視マスクとなります。なお，この外部ファイルでマスク変換をした場合，作業画面は黒で初期化されます。

●MAGIC WAND(WAND.X)

類似した色の周囲をマスク(アナログマスクが使用できる環境ではマスク値16)で囲みます。パラメータは2つで，ひとつ目は色相のしきい値，2つ目は飽和度・明度のしきい値を表し，共に0～9の範囲で指定します。色相のしきい値は大きいほど広い色相を同色とみなしますが，飽和度・明度のしきい値は色相のない色（白・黒・灰）の範囲を示し，大きいほどその他の色も同色とみなします。

実行すると図11のようなウィンドウが開き，画面左上に四角い枠が表示されます。ウィンドウの内部にはこの枠内が拡大表示されますので，囲いたいものとその外との境界付近でクリックして枠を移動させて，境界部分を拡大してください。その後，ウィンドウ内の囲いたいものの内部をクリックすると，その点を端点とする線分が表示されますので，その線分を外側に向けてクリックしてください。運がよければちゃんと囲んでくれるでしょう。もし思いどおりにならない場合はパラメータを適当に変えてみてください（多少の経験が必要です)。また，このウィンドウ上からマスクのペイント（不可視マスク）とクリア，および前で説明したMASK.Xを起動できます。

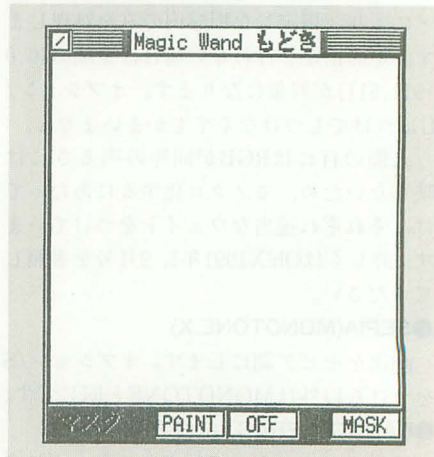
TOOL階層

●アスペクト比(ASPECT.X)

図8 拡大・縮小

図9 マスクウィンドウ

図10 アナログマスクの指定

図11 MAGIC WAND

512×512ドットモードではアスペクト比(ピクセルの縦横比）が1：1でなくて困る，というのはよくいわれていることです。X68000で65536色を発色させようと思うと，このモードでしかメーカーは保証していません。

MATIERは，コントローラを直接操作することにより，768×512ドットモードでも動作するようになっており，このモードではアスペクト比はほぼ1：1になります(あくまでもモードであって，描画領域が広くなるわけではない)。しかし，Z's STAFFではこのような機能はありません。一見簡単なように思えますが（実際に簡単なのですが)，これにはハードウェア的な制限があるのです。それは，768×512ドットモードではスプライトが使えないということです。Z's STAFFは画面取り込みなどの関係上，マウスカーソルや一部のウィンドウをスプライトで描画しているのです。このため，768×512ドットモードでZ's STAFFを動作させることはできません。

そこで，前置きが長くなりましたが，せめて確認だけでもできるようにということから作られたのがこの外部ファイルです。実行すると，マウスカーソルが消えて，768×512ドットモードになり，アスペクト比がほぼ1：1になりますので，確認をしたあと，マウスのボタンをクリックしてください。

図12 パレットウィンドウ

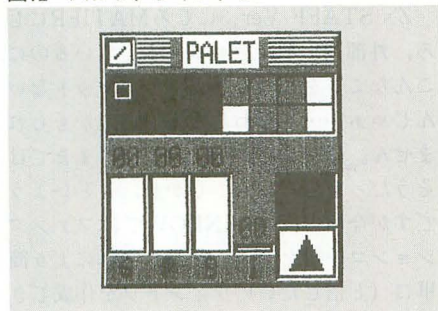
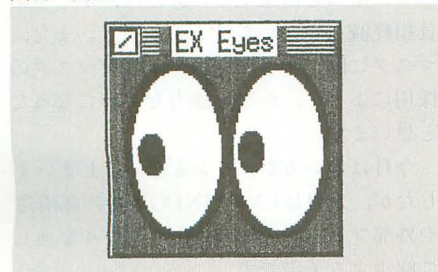



図13 目玉

●パレット(PALET.X)

いっていませんでしたが，EX-WINDOWは内部に16個のパレットを持っています。この外部ファイルは内部のパレットに色を設定したり，選択したりするものです。利用法としては，ひとつの外部ファイルで複数の色を指定したい場合や，複数の外部ファイルで同じ色を使用したい場合などに有用でしょう。今回の外部ファイルでは，CGRADで利用しています。

図12のようなウィンドウが現れますので，パレットをクリックして選択したのち，レベルバーを引っ張ってカラーを設定してください。また，▲アイコンをクリックするか，もしくはマウスを右クリックするとマウスがスポイトとなり，画面上の色を拾うことができます。

●目玉(EXEYES.X)

おまけです。シングルウィンドウ，シングルタスクのEX-WINDOWではあまり意味をなしません。サンプルがなかったので，力技で描いています。あまり参考にしないように。ちなみにウィンドウの右下を引っ張ると大きさを変えられます。

●AMI(AMIEX.X)

福嶋章太氏によるアニメーションマルチイメージAMIのローダー/セーバーです。AMIについての詳しい説明は福嶋氏の記事をご覧ください。

AMIEXを起動すると，図14のようなウィンドウが現れます。タイトルバーの下，左にあるのがファイルウィンドウ，右がカレントコマナンバーです。さらにその下の左が再生モード，右が現在編集中のAMIファイルの最大コマ数です。

ファイルウィンドウをクリックすると，

Z's-EXの秘密

本文で説明した以外にも，実はZ's-EXにはMATIER-EXにはないいくつかの機能があります。いくつかの割り込みをトラップしてあることは本文中でも述べましたが，エラー割り込みのほかにもインタラプト割り込みもトラップしてあります。これは割り込みが発生すると，待避画面からGRAMを復元して，直ちに親のプロセスに戻ります。たとえば，外部ファイル実行中にインタラプトキーを押せばZ's-EX（またはZ's STAFF）に復帰し，Z's-EXの動作中であればOS（もしくはZ's-EXを起動したプロセス）に戻ります。

ただし，この機能はあくまでも開発用の補助的な機能であり，むやみに使用するべきではありません。というのは，その後に100%の正常な動作を期待できないからです（私の能力不足です，すみません)。たとえば，インタラプトによりZ's-EX（もしくはZ's STAFF）を終了した場合は，ヒストリ機能の一部の動作に異常をきたすことがわかっています（一度無効にしてから再び有効にすると正常に動作するようになります)。

本来このような欠陥のある機能は隠し機能にするか，もしくはばっさり切り捨ててしまうべきでしょうが，外部ファイル開発においての能率に格段の差が出ますので，あえて記載することにしました（といって免責を勝ち取ろうとしている)。

また，CTRLキー＋起動キーで特定の外部ファイルを起動できるのもZ's-EXだけですし，本文にはまったく書いてありませんが，前回別の階層からESCキーでZ's-EXを抜けた場合でも，SHIFT＋起動キーで必ずルートウィンドウが開くようになっています。この起動キー関係の機能は，OPT.1＋起動キーと同様，MATIER-EXに付けるのをすっかり忘れていた機能です。次回にはMATIER-EXにもついていることでしょう。

少し話が変わりますが，Z's-EXは起動時にタイトルバーを自動的に消します。これはタイトルバーが表示されているかどうかを認識しているわけではなく，Z's STAFFが起動されてからスペースキーを押された回数を数えているだけなのです。したがって，スペースキーが押されることなしにタイトルバーの状態が変わってしまうと，その後EX-WINDOWの起動時には必ずタイトルバーが表示されるという間抜けなことになります。スペースキーを押さずにタイトルバーの状態が変わることって……実はあるんです。TELOPで取り込みを実行すると，取り込みのウィンドウを残してほかのウィンドウはすべて閉じますよね。その後，取り込みのウィンドウを閉じると，以前の状態に関わらず必ずタイトルバーが表示されるのです。つまり，取り込み実行時にタイトルバーが表示されていない状態だと，スペースキーを押さずにタイトルバーの状態を変えることになってしまうのです。ですから取り込み前にはタイトルバーを表示しておくようにしましょう。

これはZ's-EXではなくZ's STAFFの問題なのですが，現時点でZ's STAFFはASK3に正式に対応していません。以前はASKをトラップして，自前の変換ウィンドウを表示していたのですが，トラップに失敗しているのかASK3になってからASK側が行うようになってしまいました。一見問題なく動作しているように見えますが，なにかの拍子にごくまれにテキストVRAMを壊してしまうことがあるようなのです。

いってありませんでしたが，Z's STAFF（Z's-EX）はこのテキストVRAMを待避画面に使用していますので，テキストVRAMの破壊は致命的です。使うなとはいいませんが，各自の責任のもとで使用しましょう。

ファイラーが表示されます。このファイラーはPICFILERなどと操作方法は同じですので，すでにあるファイルであれば'LOAD'，新しく編集する(既存のファイルを最初から編集しなおす）のであれば'SAVE'によりファイルを選択します。ただし，'SAVE' の場合にはあらかじめ再生モード，同期カウンタを設定しておいてください(あとから変更はできません)。ファイル名を指定したあと，以下のボタンをクリックすることで，操作を行えます。

「表示」

カレントコマを表示します。

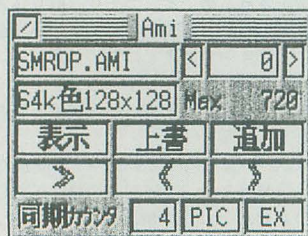
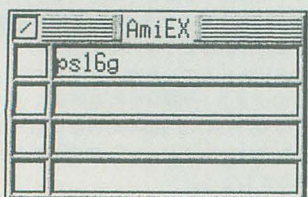
「上書き」

現在の作業画面をカレントコマに上書きします。ただし，作業画面が設定されている再生モードの色数以下に減色されていなければなりません。これについてはあとで述べます。

「追加」

AMIファイルにフレームを追加し，現在の作業画面を追加したフレームの最初のコマに書き込みます（フレームとコマの関係についてはAMIの記事をご覧ください)。色数の制限については上書きと同様です。

「≫」

AMIファイルを再生します。ただし，Z's-EXから起動した場合は，再生することによって再生前の作業画面がクリアされてしまいます。これはAMI側の問題で，福嶋氏に修正版を作ってもらおうと思ったのですが，時間がなくて今回はそのままとなってしまいました。

「《」

カレントコマを1コマ戻し，画面に表示します。

「》」

カレントコマを1コマ送り，画面に表示します。

そのほか，このウィンドウからPICFILERや任意のコマンド（EX-WINDOW専用外部ではない）を起動できるようにもしてあります。PICFILERはいいとして，コマンドを実行できるようにしたのは，前述のとおり減色を行う必要があったからです。

AMIはセーブの際に512×512から指定サイズの縮小はやってくれますが，減色処理はやってくれません。そこで，減色に関してはほかでやってもらおうというわけです。幸い巷にはいろいろな減色ツールが出回っていますので，テキストVRAMを破壊しないものであれば，そのまま使えるでしょう。他力本願ですが，時間もないことですし，私が作るよりも強力なものがあるでしょう。固定パレットモードについては，2，4，16，256色のパレットをそれぞれAMIEX.Xと同じディレクトリにあるMONO.PAL,G2T4.PAL,TONE16.PAL,G3R3B2.PALを参照するようにしてあります。

で，コマンドの実行方法ですが，ウィンドウの右下のEXボタンをクリックすると，図16のようなウィンドウが開きます。4つ並んでいる，右側の細長いボックスがコマンドウィンドウです。マウスでクリックしてキーボードからコマンドを入力してください。リターンで確定，ESCで取り消します。最大255文字までで，ボックスサイズを超えると横にスクロールします。ただし，全角文字についてはあまり真面目に対応していません。

コマンドは4つまで登録することができ，登録されたコマンドはAMIEX.Xと同じディレクトリのAMIEX.DEFというファイルに保存されます。登録が済んだら，実行したいコマンドウィンドウの左にあるボックスをクリックしてください。そのコマンドが悪いことをしていなければ，ちゃんと動くはずです。ただし，Z's-EXから起動した場合は，ものによってはテキストが表示されてしまい，美しくないかもしれませんが我慢してください。

コマンド実行後は，AMIEX側がテキストを非表示にし，画面モードが違う場合は65536色モードに変換します(実画面1024の場合は，左上512×512のみ)。

AMIの対象はSCSI装置（実質MO）のみですので，MO上のドライブのファイルの編集しか行うことはできません（ファイラーのドライブの自動認識はしません)。

図14 AMIEX

図16 外部ファイルの設定

AmiEX

ps16g

図15 再生モードの指定

Ami Mode

2色 128x128	256色128x128
2色 256x256	256色256x256
2色 512x512	64k色 64x 64
4色 128x128	64k色128x128
4色 256x256	64k色256x256
4色 512x512	16変 128x127
16色 64x 64	16変 256x255
16色 128x128	256変128x124
16色 256x256	256変256x254
256色 64x 64	

SAMPLE階層

この階層の外部ファイルは，来月号の「外部ファイルの作成」で解説する予定です。

EX-WINDOWに未来はあるか

Z's STAFF Ver.3にしろMATIERにしろ，外部ファイルをサポートしているのに，こんなことをしてもあんまりメリットないんじゃぁ……と思われる方もいるかもしれません。確かにZ's-EX ver.1.1のままではそうだったでしょう。しかし，くどいようですが今回のEX-WINDOWではファンクションコールをサポートし，それにより簡単に（と信じたい）ウィンドウを作成できるようになりました。これはユーザーインタフェイスを構築するにあたり，かなりの負担軽減につながることでしょう。また，マスクに関しても今回のアナログマスクの採用によって，かなり強力なものになったと思います。

今月は使い方がメインとなってしまいましたが，来月はEX-WINDOWの内部構造や外部ファイルの作成法をサンプルを通して解説する予定です。

X68000でモーフィングを

モーフィング画像作成ツールMorph!

Shibata Atsushi 柴田 淳

ついにX68000用モーフィングツールが完成しました。わかりやすいユーザーインタフェイスで使い勝手も良好です。このツールはDōGA CGAシステムと組み合わせて使用することで威力を発揮します。

「こちらシステムX探偵事務所」などで折りに触れて取り上げてきた，X680x0上でモーフィングを実現するツール。今回付録ディスクに収録していただく機会を得て，やっと皆さんの手元に届けられることとなった。

僕がBASICの特集記事で「多角形の最適基本図形分割」をやったのがちょうど去年の3月号のことだった。あれから1年経ったのだ。そういえばこの1年間というのは，いろいろなことがあったなあ。X68000XVI 1台しかなかったパソコンもいつのまにか3台に増えているし，のべつまくなし買いあさった本は本棚に入らないようになった。レーザープリンタとかスキャナとか，モノはこれから増えていく可能性が高いのだが，新しい機材を買っても置く場所がない。1年前はこんなにモノが増えるとは想像すらできなかった。

ところで，Macintoshにモーフィングソフトがあるのは以前も何度か触れた。ところが数カ月前，とうとうMacintoshのフリーソフトでモーフィングを実現するヤツが現れたのだ。Morpherという名のそのフリーウェアは，僕の作ったMorph!やMacの市販のモーフィングソフトと同じく，画像の上に点を打ち，それを三角形に分割して変形画像を得るというものだ。ただしMorpherの場合，点を指定したあと，人間が点を結んで三角形分割を行わなければならないのである。ひと口に「点を三角形を成すように結ぶ」といっても，プログラムで実現しようとするとなかなか難しいのだ。

こうしてみてみると，僕がしつこいくらいに追求した「自動分割」というアプローチは，あながち方向違いではなかったように思える。分割の方法も，もともと去年のBASIC特集の方法でうまくいくはずがそうはならず，ボロノイ図だとかドローネ三角形分割だとかいうコ難しい幾何学的概念まで持ち出してきたのだった。

特にドローネ三角形分割をやったときなどは，参考にしようと思っていたアルゴリズムの教科書には肝心なところで「詳細なリストを掲載することは本書の範囲を超えている」なんて書いてある始末。締め切りの3日前までアルゴリズムが思いつかず，編集部からの帰り，道に散らばる落葉を見ながらやっとうまい方法を思いついたのだ。けっこう大変だったが，まあそれなりに勉強にもなった。

モーフィングの原理についてはさんざん触れているので，ここではサラリとおさらいする程度にとどめたい。まず，変形元と変形先の2つの画像の上に，対応する点の対を置いていく。ただし，口もとなら口もと，目尻なら目尻と，点は似たような「部分」にそれぞれ置かれていなければならない。

そしてすべての点を登録し終わると，画像上に散らばった点を結んで画像を三角形に分割し，その三角形の頂点の座標を変形先のものから変形元のものへと直線移動させる。変形元の画像と先の画像から得た三角形の領域を，移動中の三角形に張りつけ，色を補間すればモーフィング画像のできあがりである。

点と線を対応させていく

うまく変形させるために

画像の変形といっても，Morph!では三角形の自由変形を行うだけである。画像の上に，同じように点を打ったとしても，三角形分割のされ方によっては微妙に変形画像が変わる。美しい変形画像を得るためには，Morph!の持っている癖のようなものをつかんでおくにこしたことはない。

そこで，ここではうまく変形させるためのコツのようなものを紹介したい。なお，ツールの詳しい操作方法は後ろの操作マニュアルを見てほしい。できるなら，Morph!の操作にある程度慣れてからこの記事を読んでいただくとなおよい。

さて，画像を得るための最初の作業として，変形させる2つの画像の，それぞれ対応する部分に点を打っていく。そして打った点には縫針ツールで線を引くことができる。実はこの線の結び方で，変形画像に大きな差が出てくるのだ。

結ばれた線は三角形分割に影響を与える。どういうふうに影響するかというと，縫針ツールで結ばれた線と交わる線は決して分割線として選ばれないのである。逆にいえば，分割線として選んでほしい線を縫針ツールで結べばいい。たとえば，顔を変形す

三角形分割を行ったところ

画像生成中……

るときなど，顔の輪郭には分割線を引いたほうがいいだろう。

また目のように目立つ部分は，周りを多角形として切り出したほうが変形がうまくいく。このようなことを考えつつ，縫針ツールを使うのだ。

ただし注意として，線は交差させてはいけない。線を交差させて分割を行うと，分割が破綻しておかしな画像を出力する場合がある。

同様に，画像一面を線で埋め尽くしても分割が破綻する恐れが出てくる。縫針ツールで引いた線と交差する線は決して分割線になることはないから，込みあった線の中にある点が孤立してしまい，そこだけ分割から抜け落ちてしまうのである。

また，2つある画像の片方に線を引いたとすると，当然もう一方にも線が現れる。で，片方の画像では引いた線と交差しなくても，もう一方ではツールで引いた線と交差してしまうような分割線があるとする。Morph!の三角形分割では，このような線も分割線として選ばれないようになっている。どちらか一方に引いた線が非常に込み入っている場合もやはり，中にある点が孤立してしまい，三角形分割から抜け落ちてしまうことがある。このような場合は，ハサミのツールを使って分割線を切っておくとうまい具合に三角形分割をしてくれる。

縫針ツールを使って三角形分割に制限を加える方法はこれでわかっていただけただろうか。目安としては，ツールで引いた線から元の画像を想像できればいい。

また線を引くこと以外にも，効果的な分割を導くような方法がある。たとえば顔対顔の変形の場合，頬の突起やこめかみなど，多角形で囲うほど象徴的でない部分には点を打ってみるといい。線で囲われた領域の中に点を打つことによって三角形分割がより細かくなり，きれいな出力を得られる。

「システムX探偵事務所」でドローネ三角形分割を取り上げたときに触れたが，この三角形分割では隣りあっている点同士を結んでいく。この分割原理を頭に置いて画像の上に点を置いていけば，かなり質の高いモーフィング画像を得ることができるはずだ。

画像出力からアニメーションまで

点を打ち，分割線として必要な線同士を結ぶ作業が終わると，次は機能メニューから「三角形分割」を選び，自動分割を始める。この分割には少々時間がかかる。かかる時間は登録した点の数に比例し，それなりのモーフィング画像を得ようと思ったら30ほどの点を登録することになるだろう。この点数を三角形に分割すると，10MHzのX68000で5，6分ほどかかる。

なお，分割に際して画像の周囲に10個ほどの点が登録される。これは画像すべてをおおうような三角形分割を得るためである。したがって，点を登録する時点では画像の周りには点は置かなくてもよい。

また，分割が終わってからも，点の移動ができるようになっている。分割された線は点線で表示されるので，これを見ながら点を動かせば，自動的に出力された分割を，人間側の思ったようなものにある程度近づけられるだろう。

分割が終わると，その情報をもとにして，いよいよモーフィング画像を生成できるようになる。メニューから「画像生成」を選ぶと設定ウィンドウが開くので，変形に要するフレーム数などを設定する。ウィンドウ内に2つのスイッチがあるが，このうち「変形のみ」とタイトルのついたスイッチをONにすると，変形元の画像と変形先の画像をオーバーラップさせず，単なる変形画像を出力する。

もうひとつのボタン，「高画質」をONにすると，誤差の少ない三角形の自由変形を使いモーフィング画像を出力する。このスイッチをOFFにするとその分画像の変形速度が速くなるが，質は格段に落ちる。時間をかけて変形させる前のプレビューとして使っていただきたい。

ところで，Morph!の出力する画像にはPICという拡張子がついているが，これはDōGAのPIC画像である。これをアニメーションとして見るためには，当然DōGAのシステムが必要だ。

まず，MKTCH.Xなどでタイムチャート

あなたもスキャナがほしくなる(サンプルデータの使い方)

ディスクに収録されたサンプルの使い方を解説しておきましょう。なお，ディスク容量の都合からサンプル画像の背景は消去してあります。

まず，コマンドラインから，

```
A>MORPH!
```

のようにしてMorph!を起動します。

最初に変形元となる右画像と左画像をそれぞれ指定して画面にロードします。「ファイル」のメニューからそれぞれの項目を選択してください。

表示されたファイル名をクリックして実行ボタンをクリックするか，直接ファイル名をダブルクリックするとロードが始まります。

次にそれぞれの画像の対応する点を指示していきます。「ピン」「縫針」「手」などのメニューを使用します。片方の絵でだいたい指示しておいて，逆側の絵の点を対応する場所につまんで移動していきます。適当に点を配置したら「機能」メニューから三角形分割を実行します。

と，ここまでの作業はすでに行った状態でセーブされていますので，点の指定がうまくできないという人はファイルメニューの情報ロードで"OTOG.POL"をダブルクリックしてください。

あとは画像生成です。「機能」から画像生成を選択します。とりあえず最初は「高画質」スイッチは入れないほうがいいでしょう。

ちなみに，生成される画像は20枚のとき合計1168Kバイト（高画質OFF）になります。データの生成先を確認しておきましょう。

ファイル名tmp???の場合なら，

```
.timechart
    tmp [1-20]
    tmp [20-1]
.endchart
```

のようなo2g.tchを作成し，

```
A>hanim -m2 B:o2g
```

のようにして実行してください。ちなみに，hanimの"-m2"は65536色データをそのまま再生するモードです。通常は，CRD.Xを使って256色データに変換したほうがよいでしょう。その場合，

```
A>CRD B:tmp
```

のようになります。変換したデータが大きすぎてhanimで実行できないときは，hanimm.xを使用してください。

綺麗なモーフィングを行うコツとしては，無理のない対応を心がけること，そのような画像を選ぶことです。全然別のものにするのもいいのですが，自然な変形は至難の技です。

ファイルを作る。DōGAのPICファイルが”TMP???”（?には数字が入る）だとすると，

MKTCH TMP

とすればタイムチャートファイルを作ってくれる。

また，モーフィングされた画像はオーバーラップを経ているため，概して色数が多く，HANIMの規定の色数上限である256色を超えてしまう。そこでアニメーションを再生する前に，CRD.Xを使い，

CRD TMP

のようにして画像の色数を減らしておく。

その後，HANIMを使ってファイルをメモリに読み込み，アニメーションを再生するのだが，その場合は先ほどのタイムチャートファイルを指定し，

HANIM　TMP.TCH

とすればいい。

なお，サンプルとしてつけたデータには分割情報も含まれている。ロードすれば，そのまま画像出力ができる。

最後に

しかし今月は恐ろしい月だった。なにしろ付録ディスクが2つも同時にやってきたのだ。引き受けた僕が悪いといえばそれまでだが，おかげで正月もずっとモニタに向かっていた。しかし68000系のC言語には触れたことがあるとはいえ，Macintosh歴たった4カ月の僕にプログラミングをさせる某Macintosh雑誌編集長もなかなか根性が座っている。それくらいでなければ，最後発で雑誌を出すなんてことはできないのかもしれないなあ。

ところでその編集長さんだが，なんでも初めて買ったパソコンがMZ-80K2Eだというのだ。この話を聞いて，なんとなく親近感を覚えてしまうのは僕だけだろうか。

余談はさて置き，これで長いこと追い続けてきたモーフィングもやっと一段落着いた。なんか勢いだけでここまできたような気がするが，まあなんとかなったのだからこれでいいにちがいない。そうだそうだ，そう決めた。

Morph！簡易操作マニュアル

Morph!.X（以下Morph!）は，2枚の画像，または1枚の画像を連続的に変形させ出力し，一般にいわれるモーフィング画像を出力するソフトである。

操作方法はフルマウスオペレーションで，極力感覚的に理解しやすいものにしたつもりだ。カンのいい人ならマニュアルなど読まずに操作できてしまうかと思うが，ここには補足的な情報も記しておくので，わからないことが出てきたら読んでみてほしい。

1．ツール

Morph!では画像を変形するにあたって，まず2つの画像の対応する場所に点を打つ。ツールとは，この作業を助けるためのものである。

ツールは5つのボタンと2つの位置変更ボタンからなっている。5つのボタンは，常にどれかひとつが黒抜きになっていて，そのボタンの操作が選択されていることを表している。操作したいボタンを選ぶときは，目的のボタンの上でマウスの左ボタンをクリックする。

次に，ボタンの並びの左から機能の説明をしよう。

●ピンのツール

このツールが選択されているときにマウスを左クリックするとマウスカーソルの位置に新しい点が登録される。

●縫針のツール

このツールの働きは2つある。まず，すでに登録されている点の上を左クリックで追っていくと，その点の間に線が引かれる。また，点のない場所でマウスの左クリックをすると新しい点を登録しながら直前に選んだ点を線で結んでいく。

また，直前に選んだ点は菱形になり，この点と次に選んだ点とが線で結ばれる。直前に選んだ点を結ぶ点から解除したいときには，その点の上で左クリックをする。

●ハサミのツール

これは見たとおり結ばれた線を切るためのツール。切りたい線の上で左クリックをすると指定した線を切ることができる。

●ゴミ箱のツール

このツールは，余分な点を削除するためのもの。削除したい点の上でマウスの左ボタンをクリックする。

●手のツール

登録した点の位置を移動したいときはこのツールを使う。動かしたい点の上でマウスの左ボタンをドラッグする。

●矢印ボタン

次に並ぶ2つの矢印ボタンは，ツールの表示位置を水平方向，垂直方向に入れ換えるボタンである。このボタンを使うことによってツールボタンを操作しやすい位置に置くことができる。

2．ポップアップメニュー

点の編集以外の操作は画面上に並んだポップアップメニューに集約されている。メニューのタイトルの上で右クリックすることによりメニューが開き，選択事項が反転表示される。望む操作が反転しているときにボタンを放せば操作が選択される。

●ファイルメニュー

a.左画像読込

選択することによりファイラーが開き，ファイルを選択すると，左の変形元画像の画面にPICファイルを読み込める。なお，読み込むサイズは256×256だけである。サイズのチェックなどは一切行っていないので，事前に縮小するなりしておいていただきたい。

b.右画像読込

右の画面，変形先の画像を読み込む。

c.情報ロード

左右の画像のファイル名，登録した点の情報，分割情報など，さまざまな情報を一括して読み込む。ただし，この情報ファイルと画像ファイルは，同じパス上になければならない。

d.情報セーブ

画像ファイル名などの情報をセーブする。

●編集メニュー

a.全連結削除

登録した線をすべて削除する。ただし，三角形分割で得られた情報は残る。一度削除した線は元に戻せないので，注意が必要。

b.全点削除

登録した点，線をすべて削除する。三角形分割の情報も削除される。

●機能メニュー

a.三角形分割

登録されている点，線をもとに三角形分割をする。操作過程が画面に表示されるので，かかる時間の目安になるかもしれない。

分割途中，左クリックで分割を中止する。

b.画像生成

モーフィング画像を生成する。ただし，まだ三角形分割が行われていないなら先に三角形分割をする。

スライドボリュームで変形に要するフレーム数を決める。ちなみに1を指定すると50%の変形画像だけを出力する。

「変形のみ」のボタンを押すと，2つの画像の色平均を取らず，変形のみの画像を出力する。また「高画質」のボタンを押さないと，三角形の自由変形に誤差は多いが速度の速いものを使う。プレビューなどに利用できるだろう。

出力するファイル名を指定しないと，自動的に”TMP”が割り振られる。画像ファイルは指定されたファイル名の後ろに3桁のフレーム番号をつけたものである。

なお，画像出力中に左クリックをすると，処理を中断する。

●システムメニュー

a.終了

Morph!を終了し，OSに戻る。

●

Morph!.Xはフリーソフトである。商的行為でない限り，プログラム，ソースに関し一切制限を設けない。また，出力された画像に関しても扱いは自由である。

Animation, Multi Image System

SCSI装置を使ったアニメーション(実践編)

Fukushima Shota 福嶋 章太

AMIはMOドライブなどの外部記憶装置を利用してアニメーション再生を行うためのシステムです。無圧縮のため機種を選ばず，10MHz機でも十分なパフォーマンスを発揮できます。

1993年9月号のMO特集を覚えているでしょうか。機種別の紹介が載っていたのでMOの購入を検討していた方には，とても役立つものだったと思います。実際，私の周囲にMOユーザーが激増したのも，あの特集の頃でしたし，「Oh!Xの特集を読んでMOを買うことに決めたぜ！」という読者は，けっこう多いのではないでしょうか。

さて，そのMO特集の中に「SCSI装置を使ったアニメーション，MOANIM.X(理論編)」という記事が載っていました。その内容は，タイトルどおり，SCSI装置を使ったアニメーションシステムの解説で，アニメーションシステムのデータ構造の解説と，簡単なサンプルプログラムが載っていました（誰も覚えてないだろうなぁ)。

そして今回，ついに（やっと）そのアニメーションシステムが完成しました。完全ではありませんが，ほぼ理論編の最後で予告したとおりの仕様を実現することができましたし，いろいろと追加した機能もありまして，それなりに使えるシステムに仕上がったと思っています。

DōGAのデータをコンバートしたもの

ではさっそく，そのアニメーションシステムの解説に移ります。

名前は「Animation, Multi Image」略して「AMI」です。特徴は，IQが300で水泳とチェスが得意……，というのは冗談ですが(名前は冗談ではない)。では，本当の特徴の解説に移ります。

AMI SYSTEM

「アニメーションマルチイメージ」，はっきりいってこじつけですが，要するに，多数の再生モードを持った（マルチなイメージの）アニメーションシステムなわけです。文法的な突っ込みは勘弁してください。自慢にもなりませんが，私はこの名前を考えるのに，数日間悩みました。とほほ。

当然ですが大容量のSCSI装置を前提としています。で，そのSCSI装置から，リアルタイムにデータを読み込みながらアニメーション再生をするわけです。

大容量のSCSI装置を前提としているだけあって，なにも考えず，どかどかと磁性面（？）を消費します。そうです，無圧縮です。私の作成したアニメーションファイルで，最大のものが32Mバイトの大きさなのですが（これで1分程度の再生時間)，MOをフロッピーより狭く感じることができます（自慢にはならない？)。

無圧縮の利点もあります。まず，どんな複雑な画像データでも再生スピードの設定可能最大値(以下，最速値)には，影響が出ません。取り込みバリバリのアニメーションも，やりようによっては可能です。さらに，X680x0本体のスピードにも最速値は影響を受けません。10MHzの初代でも，ドーピング済みの030でも一緒です。さらにさらに，読み込んだデータは，そのままVRAMに展開されるので，メモリをほとんど消費せずにアニメーションを再生できます。実行ファイルの収まる40Kバイト分も空いていれば十分なわけです。

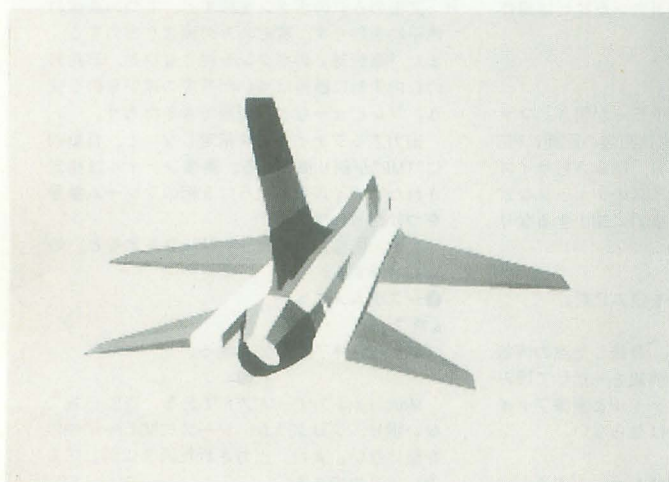
そして，これがAMIシステムの最大の特徴であり最大の売りでもあるのですが，AMIシステムの再生能力（最速値と最大再生時間)が，使用するSCSI装置に完全に依存するということです。要するに，使用するSCSI装置が速くなれば，その分AMIシステムの最速値が上がり，使用するSCSI装置が大容量になれば，その分AMIシステムの最大再生時間が長くなるわけです。

今後，SCSI装置はいま以上に高速化，大容量化が進むでしょうから，AMIシステムの再生能力は，ほっといても勝手に向上するわけです。らくちん，らくちん。

再生モード&再生能力

では，現在のSCSI装置ではどれほどの再生能力を実現することができるのでしょうか。それは，再生モードによってばらつきがあるので，まずは再生モードの解説から入りたいと思います。

現在のところ再生モードは19種をサポートしています。とりあえず表1を見てください。左から再生モード，パレット，色数，ドット数，1フレーム中のコマ数となっています。

再生モードの数値が連続していないのは，その数値を2進数で表したときに，それぞれのビットに意味を持たせるためです。各

ビットの意味については図1を見てください。空いている再生モードは，将来拡張されるかもしれません。

パレットには固定と可変とがあります。固定というのは，ひとつのアニメーションデータ全体で同一のパレットを使用することを意味し，可変というのは，1コマずつに別パレットを用意しておくことを意味します。

色数とドット数の意味はわかると思います。可変パレットモードのドット数が縦に少し潰れているのは，その部分にパレットデータが収まるためです。

1フレーム中のコマ数というのは，AMIシステム上の基本データ単位であるフレームデータ（512Kバイト固定）中に，何コマ分のデータが収まっているか，という意味です。一般にはコマ＝フレームと解釈するのが普通ですが，AMIシステムではそれらが別々の意味を持ちますので，注意してください。実際，フレームデータ中にコマデータがどのように配置されているかは，理論編で解説されていますので，そちらを参照してください。

で，やっと再生能力の話に戻ります。とりあえず私が所有するMOドライブ（SONYのRMO-S350）の最速値表を表2に示しました。このMOドライブはいまある3.5インチMOドライブのなかでは少々遅めのドライブ（信頼性は高いので，私は気に入っている）なので，機種によっては最速値がもう少し上がるでしょう（ギガ単位のハードディスクなどで試してみたいなぁ）。それでも，リアルタイムデータリードにしては，なかなかのスピードだと思います。

最大再生時間については，1フレームデータを512Kバイトとし，1フレーム中のコマ数から1コマの容量を計算し，全体の容量から全コマ数を計算し，秒間表示コマ数で割れば計算できます。

秒間表示コマ数を最速値にして，120Mバイトのデータを作ると，ほとんどのモードで，4分強の再生時間となります。

AMIで使用するデータファイル

基本的に，完成したアニメーションデータはひとつのファイルになります（通常，拡張子は［.ami］を使用）。そのほかに，パレットデータを収めるパレットファイル（通常，拡張子は［.pal］を使用），AMIの基本データ単位であるフレームデータを収めるフレームファイル（通常，拡張子は［.amf］を使用，ファイルサイズは512Kバイト固定，データ作成時に使用），などを扱います。

パレットファイルとフレームファイルに関しての注意事項があります。パレットファイルとフレームファイルには，そのファイルがどの再生モード用のデータなのかという情報は含まれていません。それでも，パレットデータなら，ファイルサイズから何色パレットなのかが判断できますが，フレームファイルは，それ単体でどの再生モードのデータなのか判断するのは非常に難しいのです（というより，どの再生モードのデータにもなり得る構造をしている)。ですから，ファイルネームを工夫するなどして各自で管理してください。

さらにもうひとつ，注意しなければならないことがあります。これはAMIファイルの最大の欠点になってしまっている問題なのですが，AMIファイルを正常に（処理落ちなく）再生しようとする場合，AMIファイルが連続セクタ上にあるという条件を満たさなくてはなりません。これは，データが不連続セクタ上に存在した場合，データの読み込み時に，シーク動作によるロスタイムが生じてしまうことに起因しています。いまのところ，これは回避不可能な条件なので，一度書いたファイルはなるべく消さないとか，フリーウェアのrefreshを持っているのなら，こまめに実行するなどの処置が必要です。

MOをお使いの方は，専用ディスクを作成し，フォーマットした直後から使用してください。ほかのファイルはなるべく同居させないほうがいいでしょう。

使用セクタが不連続なファイルが，再生できないということはありません。不連続セクタ上のファイルを再生する簡易再生モードがあるので，処理落ちはするものの一応，見ることはできます。あらかじめ，秒間表示コマ数を少なめに設定しておけば，簡易再生モードでも処理落ちなく再生することは可能です。

しかし，装置の最大の性能を発揮するためにもできるだけ専用ディスクを使うようにしてください。

図1

```
再生モードビット意味
      色数
      ↓↓↓
0 0 x x x x x x [LSB]
  ↑       ↑↑
パレット  ドット数
```

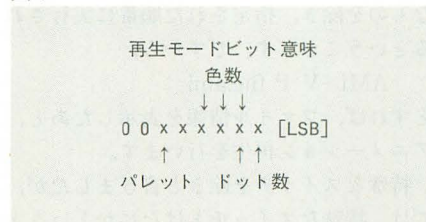



表1　再生モード表

モード(16進)	パレット	色数	ドット数	コマ数／フレーム
1 ($01)	固定	2	128×128	256
2 ($02)	固定	2	256×256	64
3 ($03)	固定	2	512×512	16
5 ($05)	固定	4	128×128	128
6 ($06)	固定	4	256×256	32
7 ($07)	固定	4	512×512	8
8 ($08)	固定	16	64×64	256
9 ($09)	固定	16	128×128	64
10 ($0a)	固定	16	256×256	16
12 ($0c)	固定	256	64×64	128
13 ($0d)	固定	256	128×128	32
14 ($0e)	固定	256	256×256	8
16 ($10)	固定	65536	64×64	64
17 ($11)	固定	65536	128×128	16
18 ($12)	固定	65536	256×256	4
41 ($29)	可変	16	128×127	64
42 ($2a)	可変	16	256×255	16
45 ($2d)	可変	256	128×124	32
46 ($2e)	可変	256	256×254	8

表2　最速値表（SONY RMO-S350）

モード(16進)	最速値［コマ/sec］	同期カウンタ
1 ($01)	60	1
2 ($02)	60	1
3 ($03)	15	4
5 ($05)	60	1
6 ($06)	30	2
7 ($07)	8.6	7
8 ($08)	60	1
9 ($09)	60	1
10 ($0a)	15	4
12 ($0c)	60	1
13 ($0d)	30	2
14 ($0e)	8.6	7
16 ($10)	60	1
17 ($11)	15	4
18 ($12)	4.3	14
41 ($29)	60	1
42 ($2a)	15	4
45 ($2d)	30	2
46 ($2e)	8.6	7

実行ファイル

●AMI.X

アニメーションの再生はもちろん，各種設定からパレットやフレームの出し入れまで，AMIファイルに対する操作はすべてこの実行ファイルで行います。

細かい操作方法はドキュメントファイルを参照してもらうとして，ここではAMI.Xの具体的な使用例をいくつか挙げてみたいと思います。

まず，

AMI -P file.ami

です。これでアニメーションの再生が行われます。そして，

AMI -V file.ami

これはアニメーションファイルの情報を表示します。

では，

AMI -P-V file.ami

これはどうでしょうか，実際に実行してみればわかるのですが，アニメーション再生を行ったあとにファイル情報を表示します。どういうことかというと，スイッチは特殊なものを除き，指定された順番に実行されるということです。ですから，

AMI -V-P file.ami

とすれば，ファイル情報を表示したあと，アニメーション再生を行います。

特殊なスイッチを除くと書きましたが，では，特殊なスイッチとはなにかというと，-?と-Mの2つです。たとえば，

AMI -P-V-? file.ami

これを実行しても，アニメーション再生や情報表示は行われません。ただ，ヘルプメッセージを表示するだけです。また，

AMI -M-P-V file.ami

これも同じで，実行結果は，再生モード表を表示するだけです。

これで，だいたいの操作方法はわかってもらえたと思います。あとは，実際にいじってみてください。

●AMIFE.X

これは，フレームデータからコマデータを出し入れするための実行ファイルです。またも，実行例で示しましょう。

AMIFE -G10,3 file.amf file.pal

これは，フレームファイルから再生モード10番のコマ番号が3のコマデータを取り出しパレットファイルに従ってVRAMに展開します。もちろん，フレームファイル，パレットファイルは再生モード10番用のデータでなければなりません。次に，

AMIFE -G18,2 file.amf

も同様にコマデータをVRAMに展開するのですが，パレットデータが指定されていません。なぜならば，65536色モードデータと可変パレットモードのデータにはパレットファイルが必要ないからです。再生モード18番は65536色モードデータです。よってパレットファイルは必要ないわけです。

ここで注意が必要です。VRAMに展開されたデータは，指定した再生モードにかかわらず，65536色モードデータに変換されます。これは，現在X680x0用の多くのグラフィックツールが65536色モード用であるということを考慮したうえでの措置です。

AMIFE -S9,4 file.amf file.pal

今度は逆にVRAMのデータをフレームファイルに書き込むわけです。VRAMへの展開と同じように，再生モードによってはパレットは必要ありません。

書き込み時も読み込み時同様VRAM上のデータは65536色モードデータでなければなりません。さらに書き込み時には，VRAM上のデータが指定した再生モードデータに変換可能な状態になっていなければならないという条件も加わります。要するに，パレットファイル内の色しか使っていないデータにしておかなければならないわけです（可変パレットの場合は使用色数だけ制限してあればよい）。

あとはAMI.X同様実際にいじってみてください。

データの制作環境

現在AMIシステム用のアニメーションデータを作る手段として考えられるのは，

1）65536色グラフィックエディタを使って1コマずつ作画する。

2）プログラミング技術と数学的頭脳と膨大な計算時間を駆使して，X680x0にアニメーションデータを作らせる。

3）豊富なデータをすでに持っているほかのアニメーションシステムからデータのコンバートをする。

以上，3点があります。

1）に関しては，とりあえず努力と根性さえあればなんとかなるとは思いますが，私は遠慮したいと思います。2）に関してはいくつかサンプルプログラムを用意しました。しかし，まだ膨大な計算時間というのが残っていますので，暇そうなX68000が近くにいたら，ぜひやらせてみてください（サンプルプログラムの説明は別記）。そして，残ったのが3）です（無理やりな展開）。

DōGAデータのコンバート

豊富なデータをすでに持っているアニメーションシステムといえば，なんといってもDōGAシステムでしょう。とりあえず，ここではDōGAのPICファイルがすでに用意してあるという前提の下で話を進めます。

コンバートなどとはいっても，やってることは単純で，1枚DōGA PICを表示してはAMIFEでフレームファイルに書き込んでいくという単純な繰り返しをバッチファ

サンプルプログラムについて

●WAVE.X&WAVE2.X

波紋をシミュレートしたAMIファイルを作成するサンプルプログラムです。WAVE.Xは基本的な波紋で光の屈折をシミュレートするプログラムで，WAVE2.XはWAVE.Xに波の反射を加え，さらに光源を定めて，その光源からの光の反射も同時にシミュレートするプログラムです。

ソースはCで書かれています。私はGCC&LIBCという環境でコンパイルしています。これ以外の環境ではおそらく，まともにコンパイルすることはできないでしょう。

実行には，それなりの時間がかかります。WAVE.XをGCCの68020&68882オプションつきでコンパイルしてX68030で実行すると40分ぐらいかかります。その他の機種では，それなりに待たされると思います。

実行するには，まず65536色のグラフィックを画面に表示させ，

WAVE file.ami

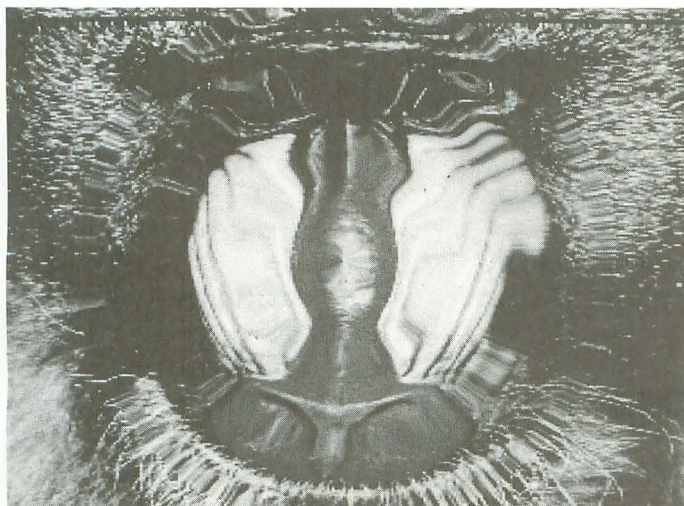
で，自動的にfile.amiが作成されます。

file.amiはプログラムをいじらない限り，32Mバイトの大きさになります。SCSI装置の空き容量には気をつけてください。WAVE2.Xの場合も実行方法は同じです。

●BOXinBOX.X

再帰図形を利用したサンプルです。

BOXinBOX file.ami

でAMIファイルが作成されます。できあがるファイルは1Mバイト程度で，実行時間も，さほどかかりはしないでしょう。

イルで行っているだけです。リスト1を見てください。いっておきますが，そのまま実行しても動く保証はありません，各自の環境にあわせて書き直したあとで実行してください。

まず，1～5行で各種初期化を行っています。AMIファイルは再生モード45で初期化しています。AMIENV.Xというのは環境変数を操作する小物ツールで，3行目で，環境変数komaに000が，4行目で，環境変数picに001がセットされます。

7～16行がメインループです。8～10行で実際のコンバートを行っています。SLIDE.XでDōGA PICを表示して，AMIZM.Xという小物ツールで，256×256ドットの画像を512×512ドットへ拡大したあと，AMIFE.Xで1コマフレームファイルに書き込みます。

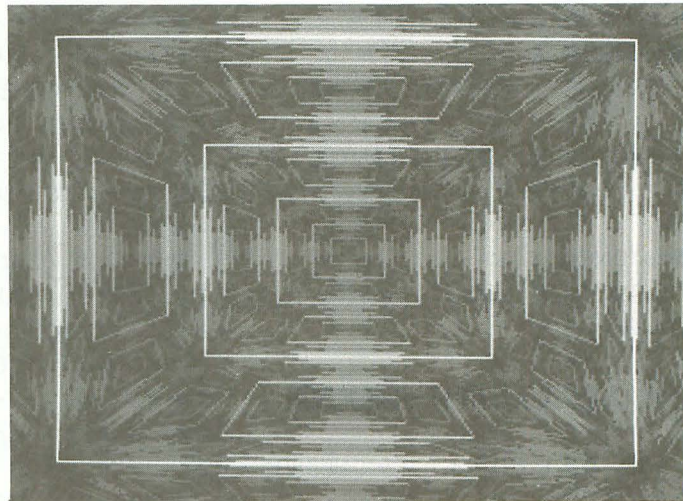
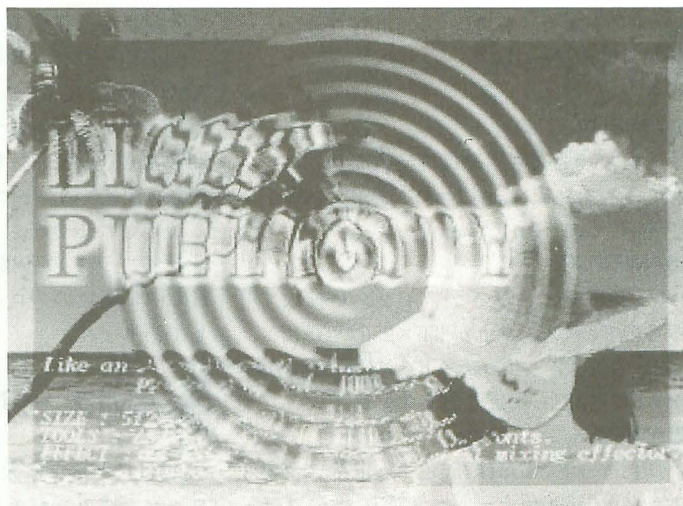
12～13行は環境変数komaをインクリメントしたあと，32(1フレーム中のコマ数)と比較して，もし32になっていたらkomaを初期値に戻し，フレームファイルをAMIファイルに書き足しています。

15～16行はループ判定です。環境変数picをDōGA PICの枚数＋1と比較しています。

18行以降はDōGA PICの枚数が32で割り切れないときの処理で，端数分を最後のDōGA PICデータで埋めています。

これで，ほとんどのDōGAデータはAMIデータにコンバートすることができると思います。

IF文など，バッチファイル用の内部コマンドに関しては，Human68kのマニュアル，SLIDE.Xに関してはDōGAのマニュアルを参照してください。

サンプルプログラムにより生成された波紋の画像。右は光源の影響をシミュレートしたもの。65536色モード全画面というAMIでいちばん重いモードを使用している。やはりこういうものは動いているところを見てもらいたい

同じくサンプル画像のBOX in BOX。再帰図形を使っただまし絵のようなアニメーションだ

リスト1

```
 1: echo off
 2: screen 1,3,1
 3: amienv -c0 koma
 4: amienv -c1 pic
 5: ami b:¥file.ami -i45,3
 6:
 7: :LOOP1
 8: slide DoGA%pic%.pic
 9: amizm -256
10: amife %temp%file.amf -s45,%koma%
11:
12: amienv -a1 koma
13: if %koma% == 032 amienv -c0 koma¦¦ami b:¥file.ami -ra %temp%fi
le.amf
14:
15: amienv -a1 pic
16: if not %pic% == 041 goto LOOP1
17:
18: if %koma% == 000 goto END
19: amienv -s1 %koma%
20: amife %temp%file.amf -g45,%koma%
21: amienv -a1 %koma%
22:
23: :LOOP2
24: amife %temp%file.amf -s45,%koma%
25:
26: amienv -a1 koma
27: if not %koma% == 032 goto LOOP2
28: ami b:¥file.ami -ra %temp%file.amf
29:
30: :END
```

小物ツール

●AMIENV.X

3桁の10進文字列を環境変数で扱うツールで，スイッチ-C[n]でnに初期化，-A[n]で+nして，-S[n]で−nします。

●AMIZM.X

65536色モードのグラフィックを拡大するもので，デフォルトが256×256から512×512への拡大，スイッチで128×128，64×64から512×512も指定できます。

●AMIPC.X

16色モード，または256色モードのグラフィックを65536色モードへ変換します。スイッチ-Pでページ指定ができます。

●toG3R3B2.X

65536色モードデータをグリーン3ビット，レッド3ビット，ブルー2ビットだけを使用した256色に減色します(ただし，グラフィックモードは65536色モードのまま)。

●toTONE16.X

toG3R3B2.Xと同様に，モノトーン16階調に減色します。

AMIシステム用ツール開発環境

一応，Cもしくはアセンブラで使用可能なライブラリを用意しました。もし，AMIシステムに少しでも興味を持って，「なにかツールでも作ってみようかな」と思った方がいましたら，ぜひともチャレンジしてみてください，よろしくお願いします。

開発用のファイルは，ライブラリ本体のAMILIB.L，C用ヘッダファイルのAMILIB.H，アセンブラ用マクロファイルのAMILIB.MACがあります。その他にはAMI.XやAMIFE.Xのソースファイルが参考になるかもしれません。

ライブラリの各関数の説明はAMILIB.Hの中に関数のプロトタイプ宣言とともに示してあります。

図2

```
$0000  ヘッダブロック
  ┌──────────────┐
  │識別シンボル    │
  │コメント        │
  │再生モード      │
  │同期カウンタ    │
  │再生フレーム数  │
  │パレット        │
  │空き            │
  └──────────────┘
$0400  フレームデータブロック
  ┌──────────────┐
  │フレームデータ  │
  │フレームデータ  │
  │フレームデータ  │
  │      ・        │
  │      ・        │
  │      ・        │
  └──────────────┘
```

AMIファイルの構造

AMIファイルは大きく分けて，2つのブロックで構成されています。それぞれ，ヘッダブロックとフレームデータブロックと呼びます。図2に簡単な図解を示しました。

AMIファイルの先頭1024バイトが，ヘッダブロックです。その内容は，識別シンボル，コメント，再生モード，同期カウンタ，再生フレーム数，パレット，空きとなっています。

識別シンボルとはそのファイルがAMIファイルかどうか識別するためのデータで，8バイトの定数で示してあります。

コメントは全角で60文字まで書くことができます。新たにデータを制作した場合，なるべくコメントをつけるようにしてください。

再生モードは，AMIファイルがどの再生モードデータなのかを示しています。

同期カウンタは，再生スピードを表すデータで，1コマを垂直同期何回分表示するかで表します。たとえば，同期カウンタが4と設定されていたら，1コマを垂直同期4回分表示するので，1秒間に15コマの表示スピードになります（垂直同期は秒間60回)。

再生フレーム数は，再生するフレーム数であって，実際にAMIファイルに含まれているフレーム数ではありませんので注意してください。

パレットは，固定パレットのデータのときに参照されます。

空きは，将来の拡張用です。通常0で埋められています。

ヘッダブロックの後ろにはフレームデータブロックがあります。フレームデータブロックはフレームデータが複数連なった構造をしています。すでに書いたように，フレームデータブロックに連なっているブロック数は，ヘッダブロックの再生フレーム数とは違いますので注意してください。

AMIシステムの今後

いまのところ，AMIシステムの内部についての拡張はあまり考えていません（唯一，PCM同期だけは早急に実現させるつもりです)。それよりもまず，外部環境の充実を優先させたいと思っています。具体的には，AMIファイルをコマ単位で編集できるツールなどです。最終的にはSX-WINDOW上で編集作業を行えるようにできたらと考えています。

まだ，できたばかりのシステムなので，バグなどが潜んでいるかもしれませんが，今後も拡張作業を続けていきますので，ご意見，ご要望，バグ報告など，編集部に送っていただけると，ありがたいです。

どうぞAMIシステムを使ってやってくださいまし。

3.5インチMOはFD化するか?

まさに逆転の発想による非圧縮アニメである。隙間商品的な色合いも強く，世間では「志が低い」ともいわれかねない。無論，ハードディスクから読み込んでVRAMへ転送……というのは誰しも考えることだろう。しかし，転送が追いつかず不可能だから，ディスク容量を食いすぎるから，といった理由で実用にはならないものと決めていた節もある。できないからみんなデータを圧縮しようとするのだ。

AMIシステムはX68000のハードウェア構成に密着していることにもよるがX68000なら比較的遅めのMOドライブを10MHz機で使った場合でも秒間512KバイトのデータをVRAMに転送できる。フルアニメどころか，真のリアルタイム再生である秒間60コマまで対応できる。ゲームなどで使うなら表示範囲が狭いワンポイント型の再生モードでもいいだろう。とりあえず実用レベルのシステムにまとまっている。

こういった発想が可能になったのもMOの普及によるところが大きい。最近は店によっては1枚2千円台のディスクも現れてきたこともあり，急速に“FD化”しつつある。

今後簡単に動画を扱える機種だって出てくるかもしれない。圧縮も課題ではある。が，68000の10MHzで特殊なハードウェアを使わずに，というとほかに手はないのだ。 (S.N.)

こちらシステムX探偵事務所

FILE-X

ピンボールを作ろう

Shibata Atsushi 柴田 淳

illustration : Y.Kawahara

なんとマスター，琴張護，春香の3人が旅行に出てしまいました。今月からは柴田氏がモーフィングに続いてピンボールの制作に挑戦するようです。しばらくは，探偵事務所を柴田氏の家に移して話が展開します。さて，どうなるでしょう。

琴張護：明日からちょっと旅行に行ってきます。しばらく探偵事務所に顔を出せませんのでよろしくお願いします。
琴張春香：あとはよろしくね！
マスター(以下M)：いきなりなんです，2人ともですか？　ひとりでいてもしようがないし，私も久しぶりにひとり旅といきましょう。じゃあ柴田君にでも電話しておきますか。
（リーンリーン）
柴田淳（以下Ats)：はい柴田です。
M：……ということで，しばらくよろしくお願いします。
Ats：そんなあ～。といっても誰かがやるんだからしばらく好きにさせてもらいますから。
（ガチャン，ツーツー）

さて，どうするかな。

そういえば，4年前，バイクに乗っていて事故にあった。一方通行の道が交差する場所で，こちらが走っていたのは優先道路。中年の女性の運転する乗用車が，一時停止を無視して僕の走っていた一方通行の道を横切り，出合いがしらにぶつかったのだ。

記憶というのは面白いもので，車がバイクにぶつかる寸前まではスローモーションのように鮮明に覚えている。そのあと，頭の中が真っ暗になり，気がついたときには乗用車と接触したところから数メートル飛ばされていた。すぐには状況がつかめず，立ち上がろうとしたが体の自由がきかない。電柱で体を打ったらしく，体の節々が痛い。左のふくらはぎには10センチほどの切れ込みがあり，血がドクドクと流れていた。

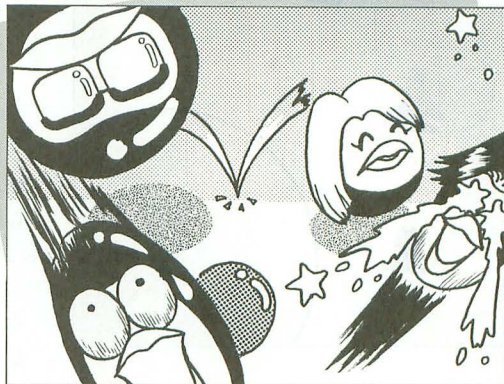
救急車で病院に運ばれる途中，救急隊員の話を聞くと，バイクと乗用車の衝突事故で僕ほどのケガで済んだというのは，かなり運のいいことらしい。このようなケースでは，打ちどころが悪く事故死，などということはざらだという。

で，僕はその話を聞いてちっともゾッとしなかった，といったら強がりに聞こえるかもしれない。しかし実際，そのときの僕は恐ろしさを感じるより，自分の死という状況をクールに想像することに忙しかった。というか，自らの死を初めて実感をもって受け止めている自分に戸惑っていたのだ。

その頃から，僕は自分の死を明確に意識するようになった。そして，ときどきこんな思考実験をしてみる。自分が順当に平均寿命まで生きたとして，死ぬのは21世紀の半ば，その頃には人類は火星にコロニーを建設しているだろうかとか，小惑星の5,6個はラグランジュポイントに引っ張ってきているだろうかとか，科学の発展に想いを巡らしつつ，人口爆発，資源枯渇など，将来の社会的問題を考える。

いまは生まれるかどうかすらわからない自分の子孫は，食料危機に直面してひもじい思いをしていないかなどと，僕の思考実験はいつも悲観的な見解にたどり着くのだが，そのあたりではたと現実に戻って現在の自分をふり返って，なんとはなしに焦燥感に駆られる。

僕はこうしてパソコンの雑誌で記事を書かせてもらっているわけだが，文章がすらすら出てくるわけでもないし，いつのまにかプログラムができあがってしまうわけでもない。こう見えても，それなりに苦しんで書いているのだ。で，このなんとはなしの焦燥感というのは，けっこう困難に向かっていく原動力になっている気がする。いつもそれなりのことをやっていないと，どうもいけない気がするのだ。

モーフィングもどうやら一段落したし，次を考える時期である。候補はいくつかあったが，以前からX680x0でゲームを作らなければならないような気がしていたので，方向はそれで決まった。しかし，普通のゲームを作るのではイカン。なにか，独特の技術を要するようなものでなければ。

そこで，ピンボールゲームを作っていきたいと思う。

反射を科学する

ピンボールを作るとはいうものの，目標にたどり着くまでには山積みの問題を解決しなければならない。特に大きな問題は，ボールの反射など，力学的運動をシミュレートするにはどうするか，というものだろう。ただしピンボールの台の上では，ボールはほぼ2次元的な挙動しか示さない。このことはいくらか明るい材料ではある。

なにが問題かを明確にするために，単純な図式を思い浮かべよう。まず，取り扱う座標を2次元に限定する。水平な板があり，角度rで飛び込んでくるボールがあるとする。このボールが板に跳ね返るとする。

図1を見てほしい。この場合，板とボールのなす角度（入射角）がrになるのはわかるだろう。そして反射したあとのボールの方向の角度（反射角）は「反対方向に入射角と同じ角度」をとる。ただ反射角の表現は2通りある。

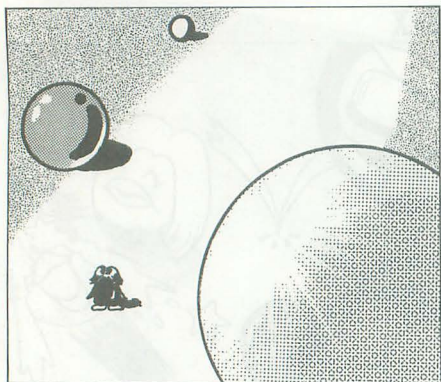
$$r1=r$$

と，rと角度の基準軸を同じに考えた，

$$r2=180-r$$

である。以降，角度については基準軸を中心に進めていく。

では次に，垂直の板にボールが反射する場合（図2）はどうか考えてみよう。こんどはボールの飛んでくる角度をそのまま入射角にはできない。たとえば45度で板に向かっていくボールがあるとすると，入射角は垂直の板とボールの軌跡のなす角度，ということになる。したがってこの場合は，入射角は－45度となるわけだ。

これを変数rを使って一般化しよう。rは基準軸とボールの飛び込む方向のなす角度

である。45度の例を思い出すと，入射角は，

r−90

となる。反射角を求めるときは，水平の板の場合を思い出してみる。水平の板の場合は，水平の角度180度から入射角を引いたのだから，同じようにすればいいのだ。つまり反射角をr2とすると，

r2＝180−(r−90)＋90＝360−r

となるわけだ。

さて次に，これを少し拡張して，板の角度を変数brにとり可変とする。この場合の入射角と反射角の関係が導き出せれば，あらゆる場合に対応する反射角の計算式が求められる。

ところで，先の板が水平な場合，br＝0になっているはずである。また，垂直の板に反射させる場合は，br＝90になっているはずだ。

このことと先の2つの式を見れば，

r2＝180＋2br−r

という式が簡単に求められる(図3)。これがあらゆる板の角度に対応した，入射角から反射角を求める式だ。と，ここまでは誰にでもできるのである。

ところで，コンピュータでボールなど動くものの座標を扱う場合，角度と速度をもとにするというのは，いかにも重そうでスマートな方法ではない。できるなら，固定小数点を使うにしろ，1回の動作で動く座標の増減，つまりベクトルで表したほうが高速な処理を行える。そこで，この反射する板の角度とボールの方向ベクトルから，反射したあとのベクトルを導く式が必要になってくる。

ここで少々頭をひねらねばならない。反射角を求める式を，

r2＝−r＋(180＋2br)

と見てみよう。するとこれは，角度−rを180＋2brだけ「回転させた」ものと見ることはできないか。ここでボールの方向を示すベクトルを(bx, by)とすると，角度−rであるからベクトルは(bx, −by)となる。ここで，高校で習う2次元の回転行列によって180＋2br回転させる。

$$回転行列＝\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

sin(180＋r)＝−sin(r)
cos(180＋r)＝−cos(r)

だから，反射したあとのボールのベクトル(bx', by')は，

bx'＝−bx×cos (2br)−by×sin (2br)
by'＝−bx×sin (2br)＋by×cos (2br)

という式で得られる。実際に反射の計算をする場合，固定小数点を使うにしろ，三角関数のテーブルから値を引っ張り出してベクトルと掛け合わせる，という手順を踏むわけだが，いったん反射後のベクトルを計算してしまえば，そのあとは足し算だけで座標値が求められる。毎回掛け算をするよりは，こちらのほうがずっとましである。

式を見るより実際動いているものを見たほうが理解が早い，という人もいるだろう。そこでBASICで組んだ簡単なプログラムを掲載しておく。ランダムな角度をとる板に，さまざまな方向からボールがぶつかっていく様子をシミュレートしてみたものだ。

さて，これで角度のわかっている板とボールの反射を計算する方法は求められたのだが，ここで新たな問題を提示しなければ

図1　水平の板にボールが反射する場合

図2　垂直の板にボールが反射する場合

図3　あらゆる板にボールが反射する場合

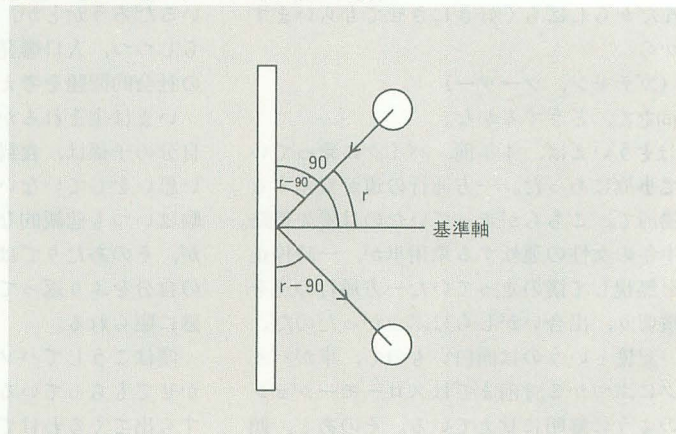
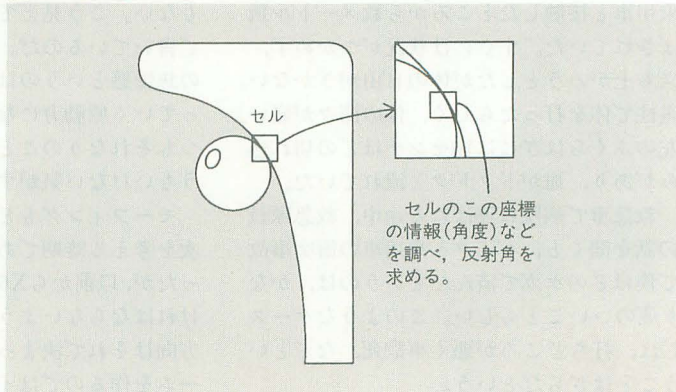



図4　セルについて

ならない。反射の計算をする以前に，壁とボールがぶつかったかどうかを調べるためには，いったいどうすればいいのだろうか。なにも真四角の箱の中でボールを弾ませるという単純なことをするのではないのだ。実際のピンボール台はいうまでもなく複雑である。その複雑さからバラエティに富んだボールの挙動が得られる，これこそピンボールの面白さの大きなファクターなのである。

セル方式と巨大テーブル

正直にいうと，これから作ろうとするプログラムの細かい仕様は，まだ詳しくは決めていない。ボールと壁の接触判定についてもそうで，いくつか有力な候補はあるが，多分それらをすべて試してみて，いちばんうまくいくものを採用することになると思う。で，その接触判定の候補に挙がっているもののひとつが「セル方式」と名づけたものである。

セルというのは，ゲームでしばしば使われる概念である。セルとはたいてい8×8の格子状をしており，これをタイルのように画面に敷き詰める。

たとえばシューティングゲームを作るとき，スクロールさせる背景に同じような絵柄が何カ所にも現れる場合がある。このような絵をベタで描き込むのはメモリの無駄遣いだ。しかし画面をセルで構成すれば，同じ画像には同じセルを使える。メモリにはどの位置にどのセルが割り当てられているかだけを記憶させればいいので，こちらのほうが効率的である。ちょうどX680x0のバックグラウンドのようなものだと思っていただければいい。

このセルという概念は，画面の絵柄だけでなく，いろいろな場面で応用がきく。同じ内容をもったデータを共有することによってメモリの消費量を減らす，という考え方なのだから，これをピンボールにおけるボールと壁との接触判定に応用する，というのはいたって自然な流れである。

具体的な処理は，まずボールの周囲が壁と接触したときにボールは跳ね返るから，描き出したボールの周囲のすべてのドットについて壁と接触していないか調べる。簡略化のため特定の1ドットだけを問題にすると，そのドットのある位置がどのセルの上に位置するかをまず調べる。そして，そのセルに該当する「接触点データ」を参照して，問題のドットが壁に接しているかどうかを判断する。もし接しているならさらに接触点の壁の角度を参照し，反射角を計算する，という流れになる。余談だが，メガドライブの「SONIC THE HEDGEHOG 2」では，主人公がピンボールばりのアクションを展開するが，内部では，おそらくここで紹介したような方法を使っているはずだ。

この方法は，メモリ消費量を低く抑えられるという利点をもっている半面，画面構成が画一的な感じになってしまうという欠点をもっている。また画面を構成するセルは「できあがりを見越して」あらかじめ用意しておくものなので，このセルの作り方には少々熟練が必要かもしれない。

この画一性を回避するため，いっそのこと1ドットごとに1バイトずつの巨大なテーブルを用意する，という方法も考えられる。512×512の画面にピンボール台を描くとして，テーブルサイズは256Kバイト。たぶん使われないであろうテキスト面をテーブルに使えば，それほど非現実的な方法ではない。面の傾きなど，せいぜい64分割されていれば十分だろうから，残った2ビットはそのほかの情報に使える。面によっては反射率が違うだろうし，またある領域に入ったらなんらかのイベントを発生させたい場合もあるだろう。それらの情報をすべてひっくるめて，1ドットごと1バイトに収めるのである。

また，テキストに256Kバイトのテーブルをとるとすると，もうひとつ分領域がとれることになる。それなら，ピンボール台を2階建ての立体構造にしてしまうというのも一興である。

反射シミュレーションの問題点

僕はかなり昔からパチンコのゲームを作ってみたいと思っていた。ぱらぱらと釘に弾かれながら落ちてくるパチンコ玉の挙動をシミュレートするというのは，なかなかプログラマの精神をかき立てる題材である。ここで紹介した方法論は，そんな経緯から随分長い間考え続けてきたものの集大成であるのだが，なにしろ現物ができあがらないので，はたしてこの方法でいいのかどうか，判断のしようがない。

特に不安なのが，ボールが壁に沿って曲がっていく運動を，先ほどの反射のシミュレーションだけで表現できるかどうかという点だ。具体的な例を挙げると，ピンボールのボールを打ち出すと，しばらくはまっすぐ進むが，台の情報で壁の曲線に沿って曲がっていく。この動きがうまく再現できるのだろうか。

この「壁の曲線に沿って曲がる」という運動は，力学的にはこう解釈できる。打ち出されたボールは，進む方向に対し少しずつ傾いた壁に次々に当たっていくのである。少しずつしか傾いていないということは「入射角が非常に小さい」ということにほかならない。入射角が小さければ反射角も小さいから，結局壁に沿って曲がっていくように見えるのだが，これも一種の反射なのである。ただし，パソコン上ではこれをデジタルに処理するのだから，誤差もあろうし，また壁の反射角の区切り方によっても動きに差が出てくるだろう。

とにかく，プログラムを組んでみて，実際にどうなるか試す以外に，確かめる方法はないらしい。そこで来月は，テーブルを用意し，ボールと壁の接触を検出する方法を検証しつつ，もう一歩踏み込んでパドルの動作をボールの動きに反映させる方法なども検討していきたい。　（つづく）

リスト

```
1000 float x,y,r,bx,by,br,xx,yy
1010 int i
1020 str s
1030 screen 2,0,1,1
1040 palet( 1,rgb(31,0,0))
1050 palet( 2,rgb(31,31,31))
1060 palet( 3,rgb( 0,31, 0))
1070 repeat
1080 wipe()
1090 r = rnd()*360
1100 x = cos(pi()/180*r)*150 : y = sin(pi()/180*r)*150
1110 line(384+x,256+y,384-x,256-y,2)
1120 bx = 384+x : by = 255+y
1130 br = rnd()*180
1140 bx = cos(pi()/180*br)*200+384
1150 by = sin(pi()/180*br)*200+255
1160 xx = cos(pi()/180*br)*5 : yy = sin(pi()/180*br)*5
1170  repeat
1180   circle( bx,by,20,0 )
1190   bx = bx - xx : by = by - yy
1200   circle( bx,by,20,1 )
1210  until (bx-384)*(bx-384)+(by-255)*(by-255) < 400
1220   x = xx
1230   xx = xx * cos(r/90*pi()) + yy * sin(r/90*pi())
1240   yy = x * sin(r/90*pi()) - yy * cos(r/90*pi())
1250  i = 0
1260  repeat
1270   circle( bx,by,20,0 )
1280   bx = bx - xx : by = by - yy
1290   circle( bx,by,20,1 )
1300  i = i + 1 : until i > 50
1310 until r = 5
1320 end
```

(で)のショートプロぱーてぃ――その54

おもろいことが大好きです

Komura Satoshi 古村 聡

春がは～やく来ないかな。寒いのはもういいぞ！ てな感じで今月のショートプロは楽しいツールやゲームが盛りだくさんです。ちょっと長めのリストもありますが，がんばって打ち込んでください。寒さに負けないで遊びましょう。

illustration：Y.Kawahara

プログラムを作るのに大事なこと。それはおもろいものを作ろうと思うことなんです。だって，おもろいプログラムなら作ってる最中は楽しいし，飽きないでしょ。うーん，我ながらなんて論理的なんだ。てーことで，プログラムを作る人は日々自分のおもろさに磨きをかけなきゃいかんわけですね。パチパチパーンチ！

ところで，その人がおもろいかどうかが如実に表れるもの。それは留守電のメッセージなんであります。

留守電って「あの機械的なテープの声を聞いた途端にさー，思わずブチッとやりたくなるんだよなー，俺」などという人が結構多いですよね。だから，おもろいこと修業中の人なんかはメッセージを入れるときに思わず「メッセージを入れてもらえますようにっ！」と力コブ入れてウケを狙ってしまうんですよ。

まあ，機械相手にしゃべると不毛な気分になるのはわかるんですけどね。ニューロでファジーな時代になって留守電が人工知能かなんかでもっと温かく，生身の人間みたいに雑談してくれるとメッセージも喜んで入れる気になるんでしょうなあ。それに人生相談できたりとか，人工知能の相手が女の子だったらうふっ，うふふふふっな会話もできるかもしれないし(そんな内容を聞かされる電話の持ち主は災難だけど)。

なんの話をしてたんだっけか？

あ，そうだ。要するになんだか私の周りの人はみんなおもろい留守電のメッセージにしてるんですよ。たとえば，Z-MUSICの西川善司は「に，に，に，西川善司です」ってラップするし，ある友人は「やぁやぁ，良い子の諸君元気かな？ ○○○マンだよ」などといいだします。これに対抗するために留守電のメッセージを変えたんです。「もしもし，私リカちゃん。お電話ありがとう～」

そ，リカちゃん電話にしたらば……誰もメッセージを入れてくれなくなった。な，なぜ，なぜなの(泣)。えーい，おもろければそれでいいんだい！ このあくなきおもろいこと探求道に栄光あれ。親が電話してきたら困るけど。

テンポアップでハイになるのだ

ではでは，めげずに今月の1本目のプログラムにいってみましょう。Z-MUSICでの曲作りに便利で，ただ遊びに使っても楽しくなっちゃうツールZTEMPO.Xです。どうぞっ！

ZTEMPO.X for X680x0
(要アセンブラ，リンカ，Z-MUSIC)
三重県 平井栄治

Z-MUSIC用のデータ，ZMSファイルを作っているときに，ある曲をコピーしてテンポを確定するために何回もEDなどでテンポを変更して面倒くさいと思ってしまったことはないですか？ 短い音が続いている部分をゆっくり演奏するのにZP－DのSHIFT＋XF3では遅すぎると思ったことはありませんか？ そんなふうに思った平井さんが作ってしまったのが，このZTEMPO.Xなのです。

このプログラムはアセンブラのソースリストの形で掲載されています。リスト1をZTEMPO.Sというファイル名でエディタで打ち込んで，アセンブル，リンクして実行ファイルZTEMPO.Xにしてください。

Z-MUSICは常駐していますね。そこでZMSファイルを演奏させてから，

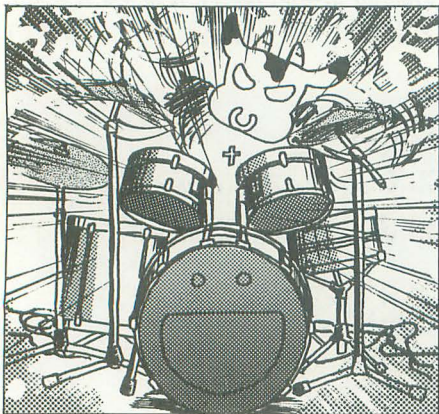
A>ZTEMPO

と実行してください。するとこんな画面が出てきます。

```
現在値            110    ↑ (＋増)
                         → (－増)
設定範囲 30～300  110    ← (－減)
                         ↓ (＋減)
```

現在値がいまのテンポ，設定範囲が設定できる範囲で，その右に表示されているのが本来のテンポです。演奏しているテンポは，カーソルキーの↑で10，→で1増やし(つまり曲を速くする)，←で1，↓で10減らす(つまり遅くする)ことができます。ESCキーでプログラムを終了します。

Z-MUSICで曲作りする人には便利なのでしょうね。私にとってはただひたすら楽しい！ 人の作ったデータを演奏させて，テンポを速くしたり遅くしたりするだけですけど。なんでしたっけ，超早回しした曲や超スロー再生した曲を演奏して，その曲のタイトルを答えさせる音楽クイズ番組があるでしょ。あれの出題者になったみたいでおもろいんですよ～。猫に小判も使いようというか……いーんだもん，私がおもろければ。さすが常連の平井さんだけあってショートプロのツボをおさえてありますね。

あ，そうだ。作者の平井さんによればZTEMPOを使うときにはテンポを(On)で設定しておいたほうがいいんだそうです。というのもTnだと[do]～[loop]内では1周するごとにリセットされるし，[do]～[loop]外でもZP－DのSHIFT＋XF4でリセットされるからだそうで，相対テンポ命令を使っている曲のテンポを変えるときには注意してください(たとえば“T＋100 T＋100T－100T－100”をテンポ100と200

で実行した場合，どちらもテンポ100となる)とのことなんです。

さーて，Z-MUSICも新しくなったことだし，こいつでいろんなデータのテンポを変えて遊んじゃおう！　ついでに(時期には早いけど)文化祭でX68000を使ってクイズなんかするとウケルかもよーん。Z-MUSICシステムver.2.0もよろしくね(宣伝宣伝)。

今月もゲームあります

さてさて，パソコンでおもろいことの代表ってーとなんといってもゲームですよね。え，先月全部ゲームだからないかと思った？　甘い甘い。てことで2本目のショートプロはBASICによるゲームプログラム，DOTMAN.BASです。どうぞ！

DOTMAN.BAS for X680x0

(X-BASIC，要VECTOR.FNC)

愛知県　水野真也

このDOTMANはBASICで書かれた逃げタイプのゲームです。実行するためには1993年9月号に掲載されたVECTOR.FNCがX-BASICに組み込まれている必要があります。

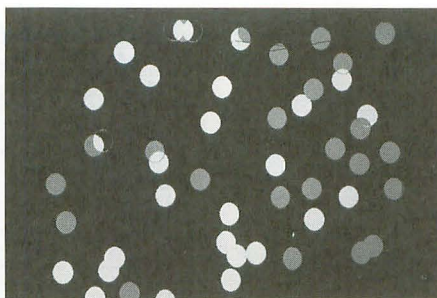
組み込み方なんですが，VECTOR.FNCをBASIC.Xと同じディレクトリにコピーします。それからBASIC.CNFに，

FUNC ＝ VECTOR

と書いてセーブします。

さて，VECTOR.FNCが組み込めたらBASICを立ちあげてリスト2を入力してください。

RUNすると画面にたくさんのマゼンタと緑の円が出てきてゲームスタートです。自分の白いドットをジョイスティックで操作して，ゴールの赤いラインまで誘導してください。緑のドットのミサイル(?)が追いかけてきます。それに当たってしまうと始めからやり直しです。緑の円の障害物にも，当たってはいけません。紫の円は安全地帯です。緑をはねかえしてくれます。

ジョイスティックのトリガーを押すと，小さな紫の円でバリアを張れます。このバリアで緑ドットを撃退できます。ただし，こっちがバリアを張ると敵も緑ドット地雷(自分の軌跡を残すだけだけど)を置いていきます。これに当たっても白ドットは死んでしまいます。

DOTMAN.BAS

この画面どっかで見たことが……そういえば，画面のなかにいっぱいゴミ(障害物なんですけど)があるでしょう。でもって敵は1ドットであっちこっちすいすい動いているでしょ……。そうか，この画面って顕微鏡を覗いたときに似てるんだ。そう考えてみると自分がミジンコか微生物にでもなった気分……。ううっ，そう思うとゴールが異様に遠く感じる～。

ま，512×512ドットの画面で自分が1ドットですから，遠くてあたりまえなのかもしれないけど。根気よく前に進んでゴールを目指しましょう。1ミリのミジンコにも0.5ミリの魂なんだな。

そして最後はばばばん！

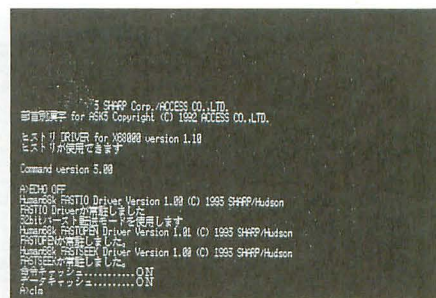
それではいよいよ今月のショートプロ最後のプログラムですね。X680x0用画面消去プログラム，CLM.Xです。どぞっ。

CLM.X for X680x0

(要アセンブラ，リンカ)

東京都　喜屋武盛道

さてさて，こいつはおもろい画面の消し方をするユーティリティですよ～ん。

このプログラムはアセンブラのソースリストの形で書かれています。エディタでリスト3をCLM.Sという名前で打ち込んで，アセンブル，リンク作業をしてください。実行ファイルの名前はCLM.Xになりますね。間違いなくできましたか？

CLM.X

さて，それではさっそく実行してみましょう。コマンドライン上で，

A>CLM

と入力してください。すると……ほ～ら，しぱぱぱぱんっ！　と画面上に表示された文字が左上から爆発して消えていくんですね～。シュールだなー，でもきれいだなー。ついでにいうと「レミングス」みたいだなーっと。

途中で待ちきれなくなった場合は，なにかキーを押すとすぐに画面をクリアしてコマンド画面に戻ります。またグラフィックRAMをRAMディスクとして使用している場合は画面のクリアのみ行い，RAMディスクは破壊しません。あ，それからFLOATn.Xは必須ですんで，必ず組み込んでおいてくださいね。

う～む。画面消去ユーティリティといえば，以前パックマンらしきものが画面を消していく「ぱっくりあ.X」が掲載されましたが，投稿自体が意外と少ないんですよ。作りやすそうなジャンルなんですけどね。

このプログラムはソースリストにコメントがいっぱいついていていいですね。たとえばソースリストの最初のほうにある，

```
HANA    equ     30
TAMA    equ     24
```

を書き換えると，画面上に同時に現れる花火の数(30)と花火1発分の火花の数(24)を変更することができるんですね。

せっかくのソースリストですから，参考にして，改造して，ついでに自分でいろいろ作ってみて，ショートプロに投稿してくださいね。それではまた来月。

リスト1　ZTEMPO.S

```
 1:                              *ZTEMPO.S 6555 93-09-25
 2:                              *ZTEMPO.R  866 12:00:00
 3:
 4:          moveq.l #$84,d0     *_B_LPEEK
 5:          lea.l   140,a1      *読み込みアドレス
 6:          trap    #15         *1ロングワード読み込み
 7:          movea.l d0,a1       *trap #3ベクタの内容
 8:          moveq.l #$84,d0     *_B_LPEEK
 9:          subq.l  #8,a1       *読み込みアドレス
10:          trap    #15         *1ロングワード読み込み
11:          cmpi.l  #'ZmuS',d0  *ZMUSIC.Xが
12:          bne     zerror      *常駐していない場合
13:          moveq.l #$83,d0     *_B_WPEEK
14:          trap    #15         *1ワード読み込み
15:          cmpi.w  #'iC',d0        *ZMUSIC.Xが
16:          bne     zerror          *常駐していない場合
17:          tst.b   (a2)            *オプションのバイト数が
18:          beq     arabia          *0の場合
19:          lea.l   kanji(pc),a6    *プログラムカウンタ相対
20:          bra     newpc           *0以外の場合
21: arabia:  lea.l   arabi(pc),a6    *プログラムカウンタ相対
22: newpc:   lea.l   new(pc),a5      *プログラムカウンタ相対
23:          lea.l   now(pc),a4      *プログラムカウンタ相対
24:          lea.l   crlf(pc),a3     *プログラムカウンタ相対
25:          lea.l   high(pc),a1     *プログラムカウンタ相対
26:          lea.l   low(pc),a0      *プログラムカウンタ相対
27:          moveq.l #30,d5          *最低値
28:          moveq.l #0,d4           *キー入力無(0)
```

```
 29:          moveq.l #$3d,d1          *get_timer_mode
 30:          trap    #3               *どのタイマで割り込み
 31:          tst.b   (a2)             *オプションのバイト数が
 32:          beq     ara300           *0の場合
 33:          move.w  #$8e4f,(a1)+     *全角三
 34:          move.l  #$819b819b,(a1)  *全角○○
 35:          tst.b   d0               *割り込みが
 36:          beq     kan077           *timer Aの場合
 37:          move.l  #$8e4f819b,(a0)  *全角三○
 38:          bra     mtempo           *timer Bの場合
 39: kan077:  move.l  #$8eb58eb5,(a0)  *全角七七
 40:          bra     low077           *timer Aの場合
 41: ara300:  move.w  #$8252,(a1)+     *全角3
 42:          move.l  #$824f824f,(a1)  *全角00
 43:          tst.b   d0               *割り込みが
 44:          beq     ara077           *timer Aの場合
 45:          move.l  #$8252824f,(a0)  *全角30
 46:          bra     mtempo           *timer Bの場合
 47: ara077:  move.l  #$82568256,(a0)  *全角77
 48: low077:  moveq.l #77,d5           *最低値
 49: mtempo:  moveq.l #5,d1            *m_tempo
 50:          moveq.l #-1,d2           *現在のテンポを返す
 51:          trap    #3               *テンポの設定
 52:          bsr     check0           *テンポを検査・修正
 53:          lea.l   old(pc),a0       *ZTEMPO.R使用前のテンポ
 54:          bsr     shiftc           *シフトJISコード変換
 55:          lea.l   (a5),a0          *現在のテンポ
 56:          bsr     shiftc           *シフトJISコード変換
 57:          bsr     vdisph           *垂直表示期間までループ
 58:          bsr     vdispl           *垂直帰線期間までループ
 59:          lea.l   shiftj(pc),a0    *シフトJISコード
 60:          move.w  #04,d3           *行カウンタ
 61: b_putb:  move.w  #22,d2           *桁カウンタ
 62: b_putc:  moveq.l #$20,d0          *_B_PUTC
 63:          move.w  (a0)+,d1         *文字コード
 64:          trap    #15              *文字表示
 65:          dbra    d2,b_putc        *桁ループ
 66:          lea.l   (a3),a1          *文字列先頭アドレス
 67:          bsr     bprint           *指定した文字列を表示
 68:          dbra    d3,b_putb        *行ループ
 69:          lea.l   range(pc),a1     *文字列先頭アドレス
 70:          bsr     bprint           *指定した文字列を表示
 71:          bsr     change           *現在のテンポを表示
 72:
 73: high00:  bsr   . vdisph           *垂直表示期間までループ
 74:          moveq.l #4,d0            *_BITSNS
 75:          move.w  #0,d1            *キーコードグループ
 76:          trap    #15              *キー入力状態のセンス
 77:          btst.l  #1,d0            *ESCが
 78:          bne     keysns           *押されている場合
 79:          moveq.l #5,d1            *m_tempo
 80:          moveq.l #-1,d2           *現在のテンポを返す
 81:          trap    #3               *テンポの設定
 82:          cmp.l   d6,d0            *現在のテンポが
 83:          beq     equal            *前のテンポと等しい場合
 84:          bsr     check0           *テンポを検査・修正
 85:          bra     nequal           *前のテンポと等しくない
 86: equal:   moveq.l #0,d7            *テンポの変化無
 87: nequal:  tst.b   d4               *キーが
 88:          bne     low000           *押されている場合
 89:          bsr     bitsns           *キー入力状態・増減値
 90:          move.b  d3,d4            *キー入力の有(?)無(0)
 91:          beq     low000           *押されていない場合
 92:          move.l  d6,d0            *前のテンポを復帰
 93:          add.w   d3,d0            *現在のテンポ+増減値
 94:          bsr     check0           *テンポを検査・修正
 95: low000:  bsr     vdispl           *垂直帰線期間までループ
 96:          tst.b   d7               *テンポの変化が
 97:          beq     low001           *無の場合
 98:          lea.l   (a5),a0          *現在のテンポ
 99:          bsr     shiftc           *シフトJISコード変換
100:          bsr     change           *現在のテンポを表示
101:          moveq.l #5,d1            *m_tempo
102:          move.l  d6,d2            *テンポ
103:          trap    #3               *テンポの設定
104: low001:  tst.b   d4               *キーが
105:          beq     high00           *押されていない場合
106:          bsr     bitsns           *キー入力状態・増減値
107:          move.b  d3,d4            *キー入力の有(?)無(0)
108:          bra     high00           *無限ループ
109:
110: check0:  moveq.l #1,d7            *テンポの変化有
111:          cmp.l   d5,d0            *テンポが
112:          bgt     check1           *最低値より大きい場合
113:          move.l  d5,d0            *テンポを修正
114: check1:  cmpi.l  #300,d0          *テンポが
115:          ble     check2           *300以下の場合
116:          move.l  #300,d0          *テンポを修正
117: check2:  move.l  d0,d6            *現在のテンポを退避
118:          rts                      *テンポを検査・修正
119:
120: shiftc:  move.w  #10,d2           *百の位を初期化
121:          move.l  d6,d0            *テンポを復帰
122:          cmpi.w  #100,d0          *テンポが
123:          blt     shiftd           *100未満の場合
124:          divu.w  #100,d0          *百の位を求める
125:          move.w  d0,d2            *百の位を退避
126:          clr.w   d0               *商をクリア
127:          swap.w  d0               *余りを得る
128: shiftd:  divu.w  #010,d0          *十の位を求める
129:          move.w  d0,d1            *十の位を退避
130:          swap.w  d0               *余りを得る
131:          add.w   d2,d2            *インデックスレジスタ
132:          add.w   d1,d1            *インデックスレジスタ
133:          add.w   d0,d0            *インデックスレジスタ
134:          move.w  (a6,d2),(a0)+    *シフトJISコードを得る
135:          move.w  (a6,d1),(a0)+    *シフトJISコードを得る
136:          move.w  (a6,d0),(a0)+    *シフトJISコードを得る
137:          rts                      *シフトJISコード変換
138:
139: vdisph:  bsr     bbpeek           *CRTCのV-DISP信号の状態
140:          beq     vdisph           *垂直帰線期間の場合
141:          rts                      *垂直表示期間までループ
142: vdispl:  bsr     bbpeek           *CRTCのV-DISP信号の状態
143:          bne     vdispl           *垂直表示期間の場合
144:          rts                      *垂直帰線期間までループ
145: bbpeek:  moveq.l #$82,d0          *_B_BPEEK
146:          lea.l   $e88001,a1       *読み込みアドレス
147:          trap    #15              *1バイト読み込み
148:          btst.l  #4,d0            *CRTCのV-DISP信号
149:          rts                      *CRTCのV-DISP信号の状態
150:
151: change:  lea.l   (a4),a1          *文字列先頭アドレス
152: bprint:  moveq.l #$21,d0          *_B_PRINT
153:          trap    #15              *文字列表示
154:          rts                      *現在のテンポを表示
155:
156: bitsns:  moveq.l #4,d0            *_BITSNS
157:          move.w  #7,d1            *キーコードグループ
158:          trap    #15              *キー入力状態のセンス
159:          moveq.l #00,d3           *レジスタをクリア
160:          btst.l  #04,d0           *↑が
161:          beq     up0010           *押されていない場合
162:          addi.w  #10,d3           *増減値+10
163: up0010:  btst.l  #05,d0           *→が
164:          beq     up0001           *押されていない場合
165:          addq.w  #01,d3           *増減値+01
166: up0001:  btst.l  #03,d0           *←が
167:          beq     down01           *押されていない場合
168:          subq.w  #01,d3           *増減値-01
169: down01:  btst.l  #06,d0           *↓が
170:          beq     down10           *押されていない場合
171:          subi.w  #10,d3           *増減値-10
172: down10:  rts                      *キー入力状態・増減値
173:
174: zerror:  lea.l   errorz(pc),a1    *文字列先頭アドレス
175:          bsr     bprint           *指定した文字列を表示
176: keysns:  moveq.l #1,d0            *_B_KEYSNS
177:          trap    #15              *キーデータバッファ状態
178:          tst.l   d0               *キーが
179:          beq     exit             *押されていない場合
180:          moveq.l #0,d0            *_B_KEYINP
181:          trap    #15              *キーデータの読み込み
182:          bra     keysns           *無限ループ
183: exit:    .dc.w   $ff00            *_EXIT
184:
185: range:   .dc.b   27,'[03A'
186:          .dc.b   27,'[10C'
187: low:     .ds.w   2
188:          .dc.b   27,'[2C'
189: high:    .ds.w   3
190:          .dc.b   27,'[2C'
191: old:     .ds.w   3
192:          .dc.b   27,'[3B',13,0
193: now:     .dc.b   27,'[05A'
194:          .dc.b   27,'[24C'
195: new:     .ds.w   3
196:          .dc.b   27,'[5B',13,0
197: errorz:  .dc.b   27,'[1;7mThe ZMUSIC.X is not included.'
198:          .dc.b   27,'[1;7m'
199: crlf:    .dc.b   13,10,0
200: kanji:   .dc.w   $819b,'一二三四五六七八九',$8140
201: arabi:   .dc.w   $824f,'123456789',$8140
202: shiftj:
203: .dc.w   '現在値      +で選択してください    ↑   (十増)'
204: .dc.w   '                                          →(一増)'
205: .dc.w   '設定範囲   ーで選択してください  ←    (一減)'
206: .dc.w   '                                        ↓   (十減)'
207: .dc.w   'カーソルキーで選択してください   ESC(終了)'
```

リスト2 DOTMAN.BAS

```
 10 /*
 20 /*          DotMan    v2.2
 30 /*                by  -Z-    1993.10.21.
 40 int px,py,x(2),y(2),xa(2),ya(2)
 50 int a,b,sp,v(2),sg ,st
 60 /*
 70 screen 2,0,1,1:console ,,0
 80 randomize(val(right$(time$,2)))
 90 /*
100 wipe()
110 for a=0 to 50
120   px=rnd()*700+50
130   py=rnd()*512
140   b=rnd()*2
150   circle(px,py,20,b*2+7).
160   paint(px,py,b*2+7)
170 next
180 line(750,0,750,512,5)
190 /*
200 px=12
210 py=rnd()*512
220 /*
230 for a=0 to 2
240   x(a)=rnd()*750
250   y(a)=rnd()*512
260   v(a)=rnd()*63
270 next
280 /*
290 while 1
300 /*
310 for a=0 to 2
320   ho=houkou(x(a),y(a),px,py,0)
```

▶「Creative Computer Music入門」ごくろうさまでした。　早野　哲也(24)香川県

```
330   sp=rnd()*20
340   v(a)=tamal(houkou2(v(a),ho))
350   if sg=0 then line(x(a),y(a),xa(a),ya(a),0)
360   xa(a)=x(a)
370   ya(a)=y(a)
380   x(a)=x(a)+vx(200-sp,v(a))/100000
390   y(a)=y(a)+vy(200-sp,v(a))/100000
400   if point(x(a),y(a))=7 then {
410     v(a)=tamal(v(a)-32)
420     }
430   line(x(a),y(a),xa(a),ya(a),9)
440 next
450 /*
460 if point(px,py)=9 then goto 100
470 st=stick(1):sg=strig(1)
480 if sg=1 then circle(px,py,3,7)
490 px=px+((st=1)+(st=7)-(st=3)-(st=9))*2+((st=4)-(st=6))*3
500 py=py+((st=9)+(st=7)-(st=3)-(st=1))*2+((st=8)-(st=2))*3
510 if point(px,py)<>9 then pset(px,py,15)
520 if px>750 then beep:end
530 /*
540 endwhile
550 /*
560 /*
570 /*
580 func tamal(a)
590   if a<0  then a=64+a
600   if a>63 then a=a-64
610   return(a)
620 endfunc
630 /*
640 func houkou2(v,ho)
650 int a
660 a=ho-v
670   if a<0 and a>-32 then v=v-3:return(v)
680   if a<0 and a<-32 then v=v+3:return(v)
690   if a>0 and a< 32 then v=v+3:return(v)
700   if a>0 and a> 32 then v=v-3
710 return(v)
720 endfunc
730 /*    誕生日     身長  体重  名前
740 /* 1993.10. ?  v1.0   1800?
750 /*          ?  v2.0   1600? ちいサイズ、スピード少しだけ良
760 /*         20  v2.1   1774  ゴールライン
770 /*         21  v2.2   2164  弾幕はり
780 /* 予定は未定  v3.0          スプライトに変更(512*512)
```

リスト3 CLM.S

```
  1: ************************************************************
  2: *        X68k CLM.X  ver 1.20      1993/12/10        By CAN     *
  3: ************************************************************
  4:
  5:         .include        iocscall.mac
  6:         .include        doscall.mac
  7:
  8: RAND    equ     $FE0E
  9: HANA    equ     30      * 30 = 同時に動く花火の数(変更可能)
 10: TAMA    equ     24      * 24 = 花火一発分の火花の数(変更可能)
 11: ********************************************  <マクロ>
 12: *
 13: * 文字ドットの有無の判定                     d5,d6 break
 14: *
 15: MOJI_D  macro   disp
 16:         move.b  disp(a3),d5
 17:         clr.b   disp(a3)
 18:         add.w   d5,d6
 19:         move.b  disp(a4),d5
 20:         clr.b   disp(a4)
 21:         add.w   d5,d6
 22:         endm
 23: *
 24: * G_PSET  X座標、Y座標、色       a0/d0,d1,d2 break
 25: *
 26: G_PSET  macro   px,py,color
 27:         local   end
 28:         move.l  #0,d0
 29:         move.l  #0,d1
 30:         move.w  px,d0
 31:         move.w  py,d1
 32:
 33:         tst.w   d0
 34:         blt     end
 35:         cmp.w   #1023,d0
 36:         bgt     end
 37:         tst.w   d1
 38:         blt     end
 39:         cmp.w   #1023,d1
 40:         bgt     end
 41:
 42:         move.w  #10,d2
 43:         lsl.l   d2,d1
 44:         add.l   d0,d1
 45:         add.l   d1,d1
 46:         lea.l   $c00000,a0
 47:         add.l   d1,a0
 48:         move.w  color,(a0)
 49: end:
 50:         endm
 51: ********************************************  <初期設定>
 52:         .text
 53:         .even
 54:
 55:         clr.l   a1              *
 56:         IOCS    _B_SUPER        * ｽｰﾊﾟｰﾊﾞｲｻﾞｰﾓｰﾄﾞへ(LOVE&PEACE)
 57:         move.l  d0,usp_save *
 58:
 59:         move.w  #-1,-(sp)
 60:         move.w  #16,-(sp)
 61:         DOS     _CONCTRL
 62:         add.l   #4,sp
 63:         move.w  d0,crt_save
 64:
 65:         clr.w   d1              *
 66:         move.w  #-1,d2          *
 67:         IOCS    _TGUSEMD        * グラフィック画面の使用状況を調べる
 68:         sub.b   #1,d0           * 0->255, 1->0 , 2->1 , 3->2
 69:         cmp.b   #1,d0           * 戻り値 d0 が1または2だったら
 70:         bls     exit            * グラフィック画面は使用不可能
 71:
 72:         IOCS    _OS_CUROF * カーソルは邪魔なので消す
 73:
 74:         move.w' #-1,d1
 75:         move.w  #-1,d2
 76:         IOCS    _B_LOCATE
 77:         move.w  d0,Yend         * 下位16ビットを取り出す
 78:         sub.w   #1,Yend         * Yend = 表示してある
 79:                                 *             テキストの最下行
 80:         move.w  #-1,d1          *
 81:         IOCS    _CRTMOD         * 現在の画面モードを調べて
 82:         cmp.w   #16,d0          * 実画面が1024*1024だったら
 83:         bge     crt_ok          * 問題ないんだけど
 84:         and.w   #$0003,d0 * <---------  ここがミソ
 85:         add.w   #$100,d0  * そうじゃなかったら実画面のみを
 86:         move.w  d0,d1     * 1024*1024にするカラクリ
 87:         IOCS    _CRTMOD   *
 88: crt_ok:
 89:         IOCS    _G_CLR_ON
 90:
 91:         lea.l   hanabi_work,a2
 92:         lea.l   $e00000,a3      * テキスト1ページめ先頭
 93:         lea.l   $e20000,a4      * テキスト2ページめ先頭
 94: ****************************************  <メインループ>
 95:         move.w  Yend(pc),d4
 96: main1:
 97:         move.w  #128-1,d3
 98: *----------------------------------------  文字の有無の判定
 99: main2:
100:         clr.w   d5
101:         clr.w   d6
102:         MOJI_D  $0        * +%00000000   例えばカーソル位
103:         MOJI_D  $80       * +%00010000  置に A が表示され
104:         MOJI_D  $100      * +%00101000  ていれば
105:         MOJI_D  $180      * +%01000100
106:         MOJI_D  $200      * +%10000010
107:         MOJI_D  $280      * +%10000010
108:         MOJI_D  $300      * +%10000010
109:         MOJI_D  $380      * +%10000010
110:         MOJI_D  $400      * +%11111110
111:         MOJI_D  $480      * +%10000010
112:         MOJI_D  $500      * +%10000010
113:         MOJI_D  $580      * +%10000010
114:         MOJI_D  $600      * +%10000010
115:         MOJI_D  $680      * +%10000010
116:         MOJI_D  $700      * +%00000000
117:         MOJI_D  $780      * +%00000000= d6.w となる
118:         tst.w   d6
119:         beq     next      * 文字が無いなら爆発をスキップ
120: *-------------------------------------  花火開花のお膳立て
121: main3:
122:         move.w  hx(pc),d6       * 開花X座標
123:         move.w  hy(pc),d7       * 開花Y座標
124: int0:
125:         move.w  #TAMA-1,d5      * 花火一発分の初期化
126: int1:
127:         G_PSET  (a2),4(a2),#0 * 最後の点を消す
128:         move.w  d6,(a2)+        * 新しいX座標
129:         dc.w    RAND            * 高速化のために、ここで求めた乱数を
130:         move.w  d0,d2           * このあと3つに分けるぞ(せこい!)
131:         and.w   #$000f,d2
132:         sub.w   #8,d2
133:         move.w  d2,(a2)+        * 新しいdX(横に飛ぶ強さ)
134:         move.w  d7,(a2)+        * 新しいY座標
135:         lsr.w   #4,d0
136:         move.w  d0,d2
137:         and.w   #$000f,d2
138:         move.w  d2,(a2)+        * 新しいdY(上に飛ぶ強さ)
139:         lsr.w   #3,d0
140:         and.w   #$000f,d0
141:         or.w    #$0001,d0       * 暗い色はよく見えないから明るい色に
142:         move.w  d0,(a2)+        * 新しい火花の色(0~15)
143:
144:         dbra    d5,int1         * 火花の数だけくりかえす
145:
146:         add.w   #1,work_count * (HANA)発分の花火のワークエリアを
147:         cmp.w   #HANA,work_count  * 使い回しているわけだ。
148:         blt     h_move          * まだワークエリアは空いている
149:
150:         move.w  #0,work_count * またNo.0のワークエリアからだ!
151:         lea.l   hanabi_work,a2  * 初期化するアドレスも戻すぞ
152: *---------------------------------------  花火を動かすルーチン
153: h_move:
154:         lea.l   hanabi_work,a1  * ワークエリアの先頭アドレスを a1 に
入れる
155:         move.w  #HANA-1,d6      * 花火の数だけループ
156: loop1:
157:         move.w  #TAMA-1,d7      * 火花の数だけループ
158: loop2:
159:         G_PSET  (a1),4(a1),#0   * 前回打った点を消す
160:         move.w  2(a1),d5
161:         add.w   d5,(a1)         * X = X + dX
162:         move.w  6(a1),d5
163:         sub.w   d5,4(a1)        * Y = Y + dY
```

▶SIDE Bはどこですか?　　嶋　真一(28)大阪府

```
164:          sub.w    #1,6(a1)           * dY = dY - 1
165:          G_PSET   (a1),4(a1),8(a1)*  X, Yの値からGRAMに点を打つ
166:
167:          lea.l    10(a1),a1
168:          dbra     d7,loop2
169:          dbra     d6,loop1
170:
171:          IOCS     _B_KEYSNS          * キー入力判定
172:          tst.w    d0                 * もしなにかキーが押されていたら
173:          bne      settle             * 後処理して終了
174:
175:          tst.w    d4                 * d4が-1なら最後のオマケループなので
176:          blt      omake              * ループ判定をスキップ
177: *-------------------------------------------   ループ判定
178: next:
179:          add.l    #1,a3
180:          add.l    #1,a4
181:          add.w    #8,hx
182:          dbra     d3,main2           * 次の半角文字をふっとばす
183:
184:          add.l    #$800-$80,a3       * 一行下がる (1ページめのアドレス)
185:          add.l    #$800-$80,a4       * 一行下がる (テキストは2ページ)
186:          clr.w    hx
187:          add.w    #16,hy
188:          dbra     d4,main1           * また一番左の文字からふっとばす
189: *-----------------------------------------   最後のオマケ
190:          move.w   #40,d3             * 40回ぐらい花火をオマケしないと
191: omake:                               * 最後の文字が消えた途端にまだ画面上に
192:          dbra     d3,h_move          * 花火が見えているのに終了してしまう。
193:                                      * でもこのあたりはスパゲッティ?(^^;
194: ******************************************   <後処理>
195: settle:
196:          IOCS     _OS_CURON          * カーソルをONにする
197:
198:          move.w   #-1,-(sp)          * 終了時にキーバッファを
199:          DOS      _KFLUSH            * クリアされたくない人は
200:          add.l    #6,sp              * この3行をコメントにすればO.K.
201: exit:
202:          move.w   crt_save,-(sp)  *
```

```
203:          move.w   #16,-(sp)          *
204:          DOS      _CONCTRL           * 画面モードを戻す
205:          add.l    #4,sp
206:
207:          move.l   usp_save,a1        *
208:          IOCS     _B_SUPER           * ユーザーモードに戻る
209:
210:          DOS      _EXIT
211: **************************************   <ワークエリア>
212:          .bss
213:          .even
214: hanabi_work:
215: ***********************************
216: * 火花一発分のワークエリア          *
217: *    .w  火花のX座標  |            *
218: *    .w        X増分  |            *
219: *    .w        Y座標  |            *
220: *    .w        Y増分  |            *
221: *    .w           色  |             *
222: ***********************************
223:          ds.w     5*TAMA*HANA     * 火花一個(5ワード)*火花の数*花火の数
224:
225:          .data
226:          .even
227: hx:
228:          dc.w     4
229: hy:
230:          dc.w     8
231: work_count:
232:          dc.w     0
233: Yend:
234:          dc.w     0
235: usp_save:
236:          dc.l     0
237: crt_save:
238:          dc.w     0
239:
240:          .end
```

ぷろぐらむ風まかせ
(3)

さーて，今月は画面出力ですよ～。先月の特集と内容がちょっと重なっちゃうような気がしますけれども……，まっ，いいか。

X68000には使える画面が3つあるんです。だからディスプレイを3台つなげてダ○イ○スができる……わけではないんです(できるとうれしいけど)。ひとつのディスプレイに写る画面がスプライトを描くための「スプライト画面」，文字を書くための「テキスト画面」，それと絵を描くための「グラフィック画面」っていう3つの画面を重ねて表示したものなんですね(図1)。

さて，このようにいろいろな画面がありますけれども，いろいろ使っていちばん楽しそうなのは……うん，グラフィック画面ですよね。なんたってお絵描きもできるし。ではさっそくグラフィック画面から見ていきましょう。

なんでも描けちゃうグラフィックなのだ！

X-BASICにはグラフィック画面に絵を描くための命令がたくさんありますよね。画面に線を描くline()，円を描くcircle()，あるいは画面に点を描くpset()とか。ところでこのグラフィック画面に字を描くための命令があるのを知っていますか？　symbol()というのがそれなんです。マニュアルのsymbol()のところを見ると，

symbol(x,y,st,h,v,mo,p,an)

x ……始点x座標
y ……始点y座標
st ……文字列(文字式)
h ……横方向の倍率
v ……縦方向の倍率
mo ……文字フォントの種類
p ……パレットコード
an ……回転角度

と書いてあるでしょ。たとえば，

symbol(0,0,"あいうえお",1,1,1,15,0)

て書けば(0,0)の座標から"あいうえお"という文字列が描かれるわけですね。

コンピュータで描いた絵をよくドット絵なんていいますけど，パソコンの画面というのは画面全体がドットという1つひとつの点が集まってできているんです。たとえば256×256ドット画面なんていいますが，あれはひとつの画面が縦256×横256，あわせて65536個の格子が集まってできているってことなんですね。線を描くのも円を描くのも，このドットを色で光らせて並べているんです(図2)。だから，ドットを文字の形に光らせるとグラフィックで文字が描ける，というわけですね。ちなみにドット1個を描く命令が先の点を描くpset()だったりします。

このドットが多いことをよく解像度が高い，あるいはレゾリューションが高いなんていいます。ハイレゾ，なんていうのはこのレゾリューションがハイ(高い)だからハイレゾリューション，縮めてハイレゾなわけですね。

X68000ではこの画面の解像度をいろいろ変えることができます。たとえば256×256ドットとか，512×512ドットとか，あるいは768×512ドットとか。その解像度を変える命令はscreen命令です。マニュアルを見るとこんなふうに書いてあります。

screen(表示画面サイズ，グラフィックの実画面サイズおよび色モード，ディスプレイ解像度，グラフィックの表示ON/OFF)

この表示画面サイズっていうのが要するに解像度でして，その引数によって，

0 ... 256×256ドット
1 ... 512×512ドット

リスト

```
100 int hx,hy,ax,ay,sx,sy,p,st
110 hx=0:hy=0:ax=0:ay=0
120 /*画面の初期化*/
130 screen 0,0,1,1:window(0,0,1023,1023)
140 home(0,hx,hy)
150 for i=1 to 1000
160   sx = rnd() * 1024
170   sy = rnd() * 1024
180   p = rnd() * 15
190   pset(sx,sy,p)
200 next
210 locate 13,8:print"Ready!"
220 while(strig(1)=0):endwhile
230 cls
240 /*メインルーチン*/
250 while(1)
260 hx=hx+ax
270 if(hx>1023) then hx=hx-1024 else if(hx<0) then hx=hx+1024
280 hy=hy+ay
290 if(hy>1023) then hy=hy-1024 else if(hy<0) then hy=hy+1024
300 home(0,hx,hy)
310 st=stick(1)
320 switch st
330   case 1:ax=ax+1:ay=ay+1:break
340   case 2:        ay=ay+1:break
350   case 3:ax=ax-1:ay=ay+1:break
360   case 4:ax=ax+1        :break
370   case 6:ax=ax-1:        break
380   case 7:ax=ax+1:ay=ay-1:break
390   case 8:        ay=ay-1:break
400   case 9:ax=ax-1:ay=ay-1:break
410 endswitch
420 if(ax>32) then ax=32 else if(ax<-32) then ax=-32
430 if(ay>32) then ay=32 else if(ay<-32) then ay=-32
440 endwhile
```

2 ... 768×512ドット

というように表示画面サイズが決まるわけです。わかったかな？

表示画面でスクロール

さて，さっきのscreenの説明のところで解像度が表示画面サイズって書いてあったわけですけど，2つ目の引数も「グラフィックの実画面サイズおよび色モード」と，サイズって文字が出てきてますね。しかもグラフィックの実画面のサイズ。う〜む，あやしい数字だ。

実はグラフィック画面というのは，ディスプレイに見えている部分だけをいうのではないのです。X68000ではグラフィックは図3のように絵のデータが描かれる実画面というものがあります。で，このうちいくらかの部分が表示画面といってディスプレイに表示されているんです。この実画面サイズや色数はscreenの2番目の引数で，

0 ... 1024×1024 16色
1 ... 512×512 16色(ページ数4)
2 ... 512×512 256色(ページ数2)
3 ... 512×512 65536色(ページ数1)

こんなふうに変えられるのです。

ページっていうのは，図3では実画面は1枚ですが，この実画面を何枚かもって使いたいページを切り換えて使う，なんてことができる機能のことなのです。たとえば先月の特集のように画面を描き換えるときに起こるちらつきを防ぐために使えますね。

ところで，実画面のなかに表示画面があるんですよね。ではこの表示画面をずらすとどうなるんでしょう。てことで作ってみたのが今月のサンプルプログラムです（リスト）。

このサンプルはX68000のグラフィックの実画面と表示画面を使った宇宙への旅です。

プログラムを入れてRUNすると画面に星が描かれて，しばらくすると画面中央に「Ready!」という字幕が出ます。そこでジョイスティックのボタンを押してから，ジョイスティックを動かすとその方向に星が流れていきます。

この星の流れていく様子を表示画面で表現しました。そう，もう気づいた方もいると思いますけれど，この表示画面をずらすことで簡単にグラフィック画面のスクロールができてしまうんです。

プログラムの解説をしましょう。最初の2行で変数の初期化をします。hx, hyは表示画面の左上が実画面のどこにあるかを示す座標です。ax, ayはx，y方向の星が流れる速度なので，それぞれ0を入れておきます。

120行からは画面の初期化をします。まず，先ほどのscreen命令で表示画面を256×256ドットに，実画面を1024×1024ドット，16色モードにします。その次のwindow命令は実画面に絵を描いてもいい範囲を決める命令です。

次は140行のhome()関数ですね。この関数がこのプログラムのいちばんのカギで，表示画面を実画面のどこから表示するかを設定する関数なのです。ここではhx,hyは0ですから，(0,0)，つまり，実画面の左上端から表示画面が始まるのですね。

で，150〜200行で1000個の星を描きます。それから210行で「Ready!」を書いて下準備は終わりです。

さて，そしてメインルーチン。while〜endwhileで無限ループになってますね。まず，ちょっと飛ばして310行から解説します。

310行はstick()関数ですね。これでジョイスティックの内容を知ります。で，そのあとのswitch〜caseで，ジョイスティックの内容によって星のx，y方向の速度を増やしたり減らしたりします。420, 430行は速度のリミッターです。if文でax, ayがあまり大きすぎたらカットするようにしているわけです。

で，endwhileで戻ってきて260行。速度ax, ayをいまの仮想画面の実画面内の座標に加えて，300行のhome()関数で新しい表示画面の座標に変えてスクロールするのです。270行では，新しいx座標，y座標が1024以上になったり0以下になってしまった場合の処理をしています。

if(hx>1023) then hx=hx-1024

実画面は0〜1023までですから，表示画面が実画面の右側にはみ出してしまった場合ですね。その場合は，もう一度左端に戻しています。

if (hx<0) then hx=hx+1024

左側からはみ出した場合は，画面の右端から表示し直すんですね。

あれ，なにか気がつきませんか？　そう，表示画面の左端が実画面の右端からはみ出すちょっとだけ前，実画面の右端から表示画面が途中からはみ出してる場合はどうなっているのでしょう。プログラムを実行してずーっと右にいっても違和感はなかったですよね？

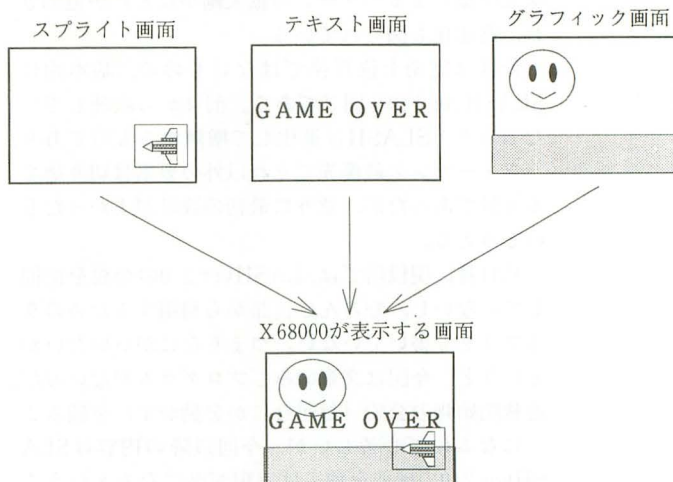
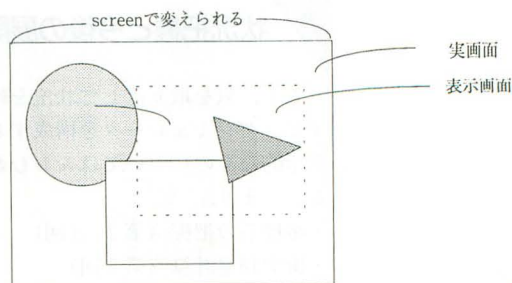
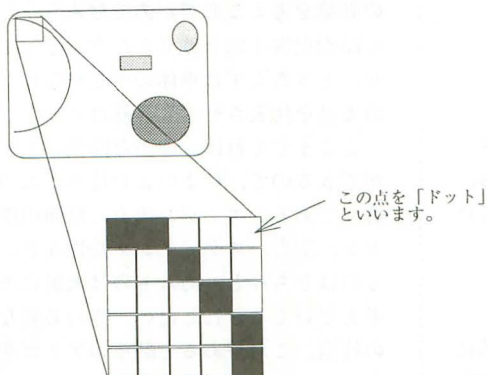
実は表示画面が実画面の右端からはみ出した場合は，そのはみ出した部分は実画面の逆側，x=0の位置から表示されるのです。同様にy方向もはみ出すと逆の側にいってしばらくすると元の位置に戻ってくる「だって地球は丸いんだもーん」という状態になっているのですね。このグラフィック画面のような仕組みを「球面スクロール」といいます。球だからどこへいってもぐるぐるまわるわけですね。

グラフィックについては以上です。来月はテキストとスプライトについてやっていきましょう。ではまた。

図1　画面の重ね合わせ

図2　拡大してみると……

図3　表示画面と実画面の関係

ハードコア3Dエクスタシー(第6回)

SIDE A

ドライビングシミュレータのためのコース構築法

Tan Akihiko 丹 明彦

**今回は，パラレルワールドでの判定，マップシステム，多段階ディテールを考え
より速く，より効率的に処理するための方法を提示していく
そして，SLASHver.2.0を手に目標を追いかける**

SLASHver.2.0完成直前記念のお詫び

横内氏の献身的な努力と実力ある読者の貢献により，SLASHver.2.0も完成間近となった。私から要望していたもの（座標変換パラメータの行列による直接指定など）をはじめとするいくつかの新機能（3次元座標によるパターンの拡大縮小など）が追加され，高速化も図られている。

仕様は完全上位互換ではないものの，基本的にSLASHver.1.0と同じである。前々から吹聴していたとおり，SLASHは進化して増殖するものであり，パフォーマンス最優先でそれ以外の要素は切り捨てる方針であったが，意外に最初の設計がよかったものとみえる。

私自身，現段階ではSLASHver.2.0の全貌を把握していないし，むろんC言語から利用するためのライブラリも書いていない。つまりなにがいいたいかというと，今回は文章のみでプログラムがないのだ。連載開始時の公約（毎回なにかを動かす）を破ることになるので心苦しいが，今回以降の内容はSLASHver.2.0の機能を使えば実現が楽になるということで，具体的なプログラムの制作はSLASHver.2.0への移行が完了してからということにする。まったくもって申しわけない。

状況把握と今後の展開

さて，気を取り直して状況を眺めてみると，ドライビングシミュレータを構成する要素技術のうち，数学的なものについてはおおむね議論が終わっている。つまり，

・座標系の把握（第2，3回）
・衝突判定計算（第5回）

を押さえているということである。ただ，座標系については，SLASHver.2.0の「行列による座標変換」の導入によって，よりすっきりとした実装が可能になるので，もう一度説明することになるだろう。

そして，これから考えていく必要があるのは，主に力学的な要素である。まず，

・より精密な路面の把握

である。これはつまり，自動車には4つのタイヤがあるが，そのそれぞれがきちんと接地したときに車体がどういう姿勢になるかということである。基本的には衝突判定計算を駆使するのだが，もはや純粋な数学的計算だけでは不足なのである。自動車にはサスペンションがついており，車体とタイヤの位置関係は常に変動する。サスペンションが柔らかければ路面に少々の凹凸があってもタイヤは接地するだろう。逆に硬いサスペンションであればちょっとしたことで1輪くらいは簡単に浮き上がる。派手なギャップでは車体もろとも飛び上がることだってあるだろう。コトはすでに自動車力学の領域に入っているのである。どの程度の力学モデルをたて，どのくらい精密なシミュレーションを行うのかということに左右される。

次に，旋回運動であるが，これは，

・水平面上の円旋回運動（第4回）

ですでに一部が議論されている。とりあえずは，

・非水平面上の円旋回運動

の計算を考える必要があるだろう。むろん，先ほどの路面把握手法を確立したうえでのことになるのだが，とりあえずは車体の中心かなにかで自動車全体の姿勢を代表させても計算はできる。

ここまでくれば，適当な路面の上で走り回ることができるので，いよいよお待ちかねの自動車力学の導入である。エンジン出力，路面の摩擦，トラクション，空力，ステアリング特性など。これらをあるものはきちんと，あるものは大胆にモデル化しつつ考えていくことにしたい。絶対必要なのは自動操縦の技術。これがあると敵車のアルゴリズムが確立できるのだ。

そしてその先はゲームとしての作り込みである。古今東西のカーレースゲームを参考にしつつ，考えうる最高のゲームシステムにしたいものだ。

いやはやなんとも壮大だ。独力でできるとはとても思えんなこりゃ。

コースのデータ構造

ということで寄り道をしているわけにはいかない。さっそく地道に1歩前進したい。お題はコースのデータ構造。

モデルとしてはレーシングカーおよび乗用車を想定する。つまり，コースはおおむね平坦。軽度のアップダウンやバンクは存在する。車も基本的に接地したまま運動し，派手にジャンプしたり宙返りしたりはしない。このことはデータ構造には特に影響を及ぼさないはずだが，プログラムを書く際にいくつか処理を省略できる。

データ構造の特徴は次の2つである。

・パラレルワールド
・マップシステム

それぞれ解説していくことにしよう。

パラレルワールド

前回に少しばかり触れたことである。

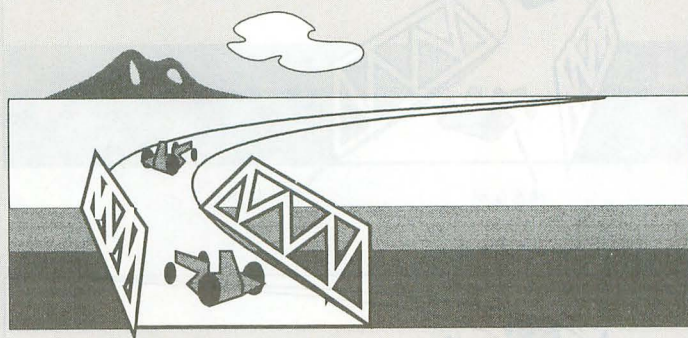
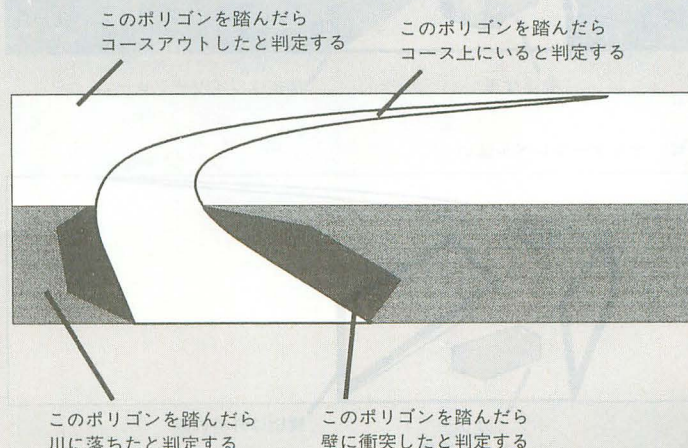
コース上をきちんと走るためには，コースを構成する各ポリゴンとの衝突判定が自由に，かつ素早くできる必要がある。それはタイヤが路面を捉まえるために使われたり，車がコースアウトしたことを判定するために使われたりする。また，ドライビングシミュレータとしての体裁を整えるためには，コースを構成するポリゴンは見栄えがよく，かつ高速に表示できるものでなくてはならない。すなわちコースを構成するポリゴンには，衝突判定と表示のそれぞれにおいて適切な性質を持っていなくてはならないという条件が課せられている。そしてその適切な性質は，衝突判定と表示において微妙に異なる。

そこで，パラレルワールドの考え方が浮上することになるのである。原理は簡単で，表示のためのポリゴン（図1(A)）と衝突判定のポリゴン（図1(B)）を別に用意しておく。両者は大まかな形も縮尺も共通だが，後者には必要最小限の情報しか入っていない。これにより，衝突判定に要する処理速度を軽減できるのである。

衝突判定用の情報としては，先月のCHECKINFO構造体かその改良版を用いる。判定関数にはポリゴンがほぼ上向きと仮定して使う関数と任意の方向のポリゴンに対応した関数があったが，処理速度の観点から前者を用いる。通常のサーキットしか扱わないのならばこれで十分であろう。

壁をどうするかはひとつの問題だ。壁というのはつまり，車がそこから先に進めないということである。ポリゴンというのは数学的な観念で，ただ定義しただけでは壁として働いてくれず，車はそこを突き抜けてしまう。壁は垂直に立っているので，使うことにした上向きポリゴン用の衝突判定関数は使えない。正攻法は，車の軌道と壁のポリゴンの交差判定を，任意の方向のポリゴン用の衝突判定関数を使って行うことである。が，なんとなく重そう。

まだ試していないのでうまくいくか確かめていないのだが，ひとつの案を示しておく。上向きポリゴン用の衝突判定関数は，車が踏んでいるポリゴンを判定できる。これを利用して，壁の陰に隠れている地面にあたる部分にポリゴンを作る。これを踏んだら壁に当たったと判定するのだ。もちろんパラレルワールドの表示しない世界のポリゴンなので，形が表示するものと極端に違っていようとかまわないことになる。

図1　パラレルワールド

さらに，表示用のポリゴンそのものにもパラレルワールドの概念を持ち込む。コースの作り込みをたとえば3段階程度にする(図2(A)(B)(C))。この3段階のディテールをもった表示用データを，一定の規則に従って使い分ける。このテクニックは，主に表示処理の負荷をコントロールして，もたつきのない画面描き替えを実現するために用いられる。

(例1)手動切り替え……ユーザーがドライビングシミュレータで遊んでいて，どうも反応が鈍いなと思ったら，ディテールを落とすという操作ができるようにする。

(例2) 最低fps数キープ……fpsとはframe per secondの略で，毎秒何フレーム描けるかという値。この値が大きいことは動きが滑らかで操作に対するレスポンスがいいということを意味する。逆に値が小さいと動きは粗くなり，操作にも支障をきたす。それを避けるために最低fps数を指定できるシステムも存在する。

たとえば最悪でも秒間10フレーム (10fps) を確保するように指定しておけば，表示する物体が多くなってきて10fpsをキープできなくなったときだけ自動的にディテールを落とすという自動制御とする。

(例3) 遠近による切り替え……一般に3Dものでは，遠くのものは見かけの大きさが小さくなるので，いたずらにディテールを上げたところで表示が潰れるだけである。形が複雑なぶん座標変換にもポリゴン描画にも手間がかかっているのに，なんとも無駄の多い話だ。

そこで，遠くにある物体を表示する際には積極的にディテールを落とすようにする。ディテールを落とした物体は大まかな形であり，表示が潰れにくい。たとえば車ならディテールを落とす場合にタイヤなしでモデリングし，ボディの色のみを強調する。また橋桁は外枠のみを定義する。

さらに副作用として，一般に3Dものでは遠くにある物体ほどたくさん存在するので，処理が軽くなることも期待される。この手法は次のマップシステムとの組み合わせで威力を発揮する。

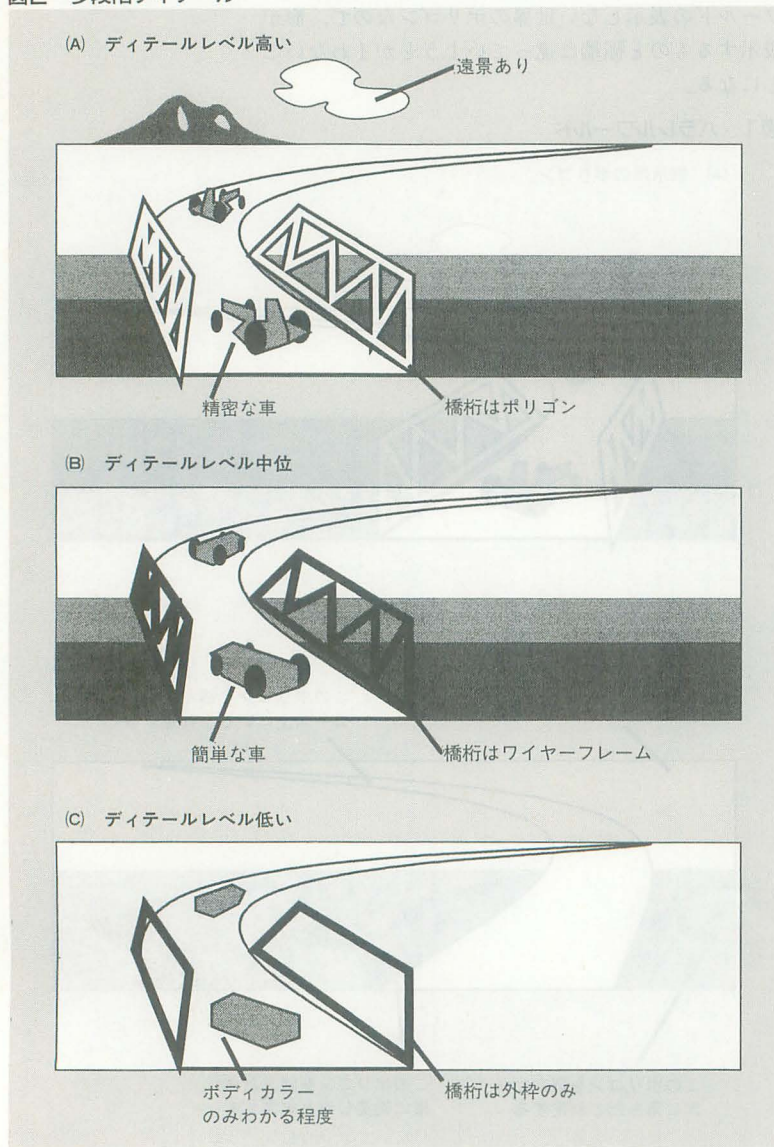
図2 多段階ディテール

マップシステム

マップシステムを導入する背景もやはり「処理速度の向上」にある。

いまさらいうまでもないことだが，一般に3Dゲームは，広大な空間の中にゲーマーが入り込み，世界のほんの一部を立体的に見ながら移動していくゲームである。「ほんの一部」というところがポイントだ。そう，本当に表示に必要な物体は，そのゲーム世界にある膨大な物体のごく一部である。だが厄介なことに，ある物体が表示されるかどうかは，その物体の座標をゲーマーの視点に従って座標変換（透視変換または透視投影）し，画面に入るかどうかを調べるまではわからない。そして，正直に全物体の座標変換を行ったところで，本当に使われるのはほんのわずかである。これは明らかにもったいない(図3(A))。

これを回避する方法はいくつか考えられる。たとえば視点から表示しようとする物体に向かうベクトルと視線のベクトルの内積を取り，符号が負なら，すなわち物体が視野に対して真横かそれより後ろにあるなら座標変換も表示もしない(図3(B))。これはそれなりに精度の高い方法である。

マップシステムはこの考え方をもう少し押し進めたものである。ゲーム世界をいくつかのブロックに分割し，表示物体が視野に入るかどうかを物体単位

でなく，ブロック単位で調べるのである。あるブロックが視野から外れていると判定された場合，そのブロックに属する物体は一切無視する。地面や建物は初期設定時に各ブロックに登録される。車はプレイ中の各瞬間にどのブロックの上にいるかを求めたうえで，そのブロックが視野から外れているかどうかをチェックする。

ブロック分割の方法としては，分割線を等間隔に取って長方形に分割する方法(図3(C))とコースの形状に沿って任意形状に分割する方法(図3(D))が考えられるが，私は前者の等間隔分割を採用することにした。地面や建物はブロックに固定されているのでどちらの方式でも計算コストはそうかからないのだが，車がどのブロックにあるかを調べるためには任意形状のブロックよりも長方形のブロックがなにかと都合がよいのだ。(x,z)座標を適当な定数で割ればブロックの番号が求められる。

衝突判定においてもマップシステムは有用だ。ゲーム世界に存在するすべての衝突判定ポリゴンをチェックせずとも，車の存在するブロックに属するポリゴンに関してだけチェックを行えばよいからだ。

余談だが，SLASHがあまり巨大なポリゴンを扱えないというのも，マップシステム採用の積極的な理由になる。広大なゲーム世界は，小さいポリゴンで構成されたブロックの集合で表現するほかない。まあ，これはSLASHの都合だ。ver.2.0でマップシステムが導入されることにより，事実上無限の大きさのゲーム世界を表現できることになるわけだ。

多段階ディテールとマップシステムの組み合わせ

さて，パラレルワールドの項で説明した多段階ディテールの視点からの遠近による使い分けは，マップシステムと併用することで実装が楽になる。つまりあるブロックからほかのブロックまでのおおまかな距離はわかっているのだから，これを利用するのだ。視点の存在するブロックから表示するブロックまでの距離から，そのブロックを表示すべきディテールの高さを求める。

図4(A)をご覧いただきたい。簡単な例として，A，B，C，Dの4つのブロックからなるコースで考える。

まず自車がAブロックにいる場合を考える。ほかの車はB，C，Dブロックにそれぞれいる。Bブロックにいる車と橋桁がいちばん近いので詳しく描画するだろうし，C，Dブロックとなるにつれて遠くなるのでディテールを落とす(図4(B))。

次に自車がBブロックにいる場合を考える。すると，Cブロックは詳しく，Dブロックはやや簡単に描画すればよい。そしてAブロックについては，自

図3 マップシステム

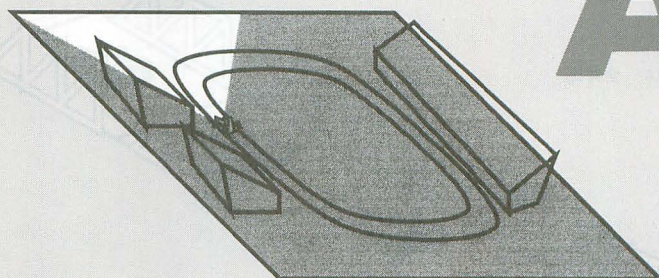
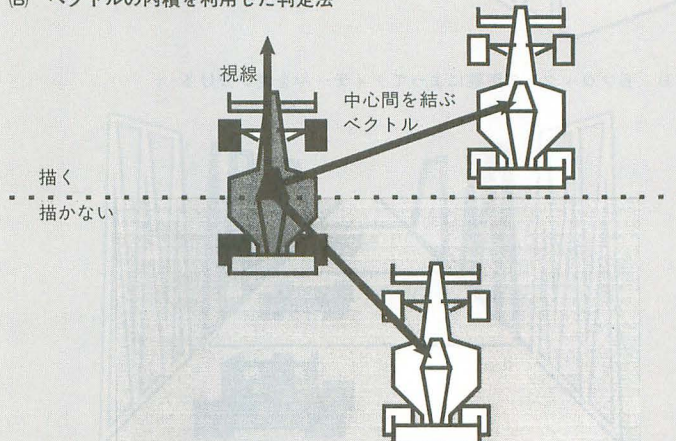
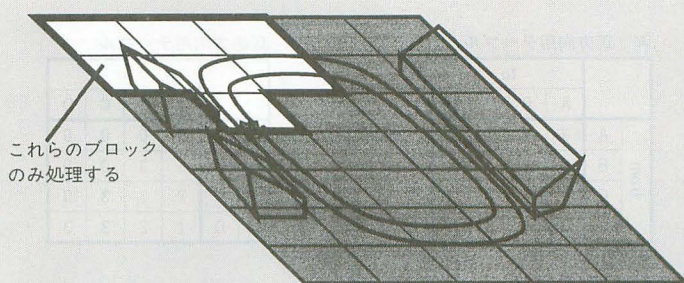
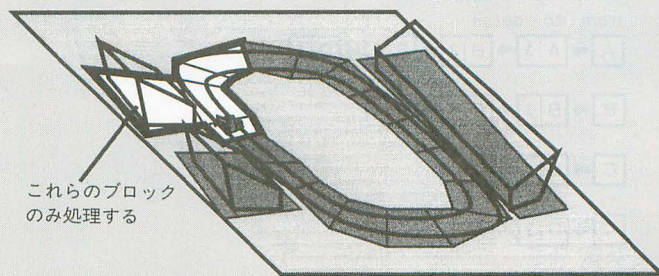



図4　多段階ディテールとマップシステムの併用

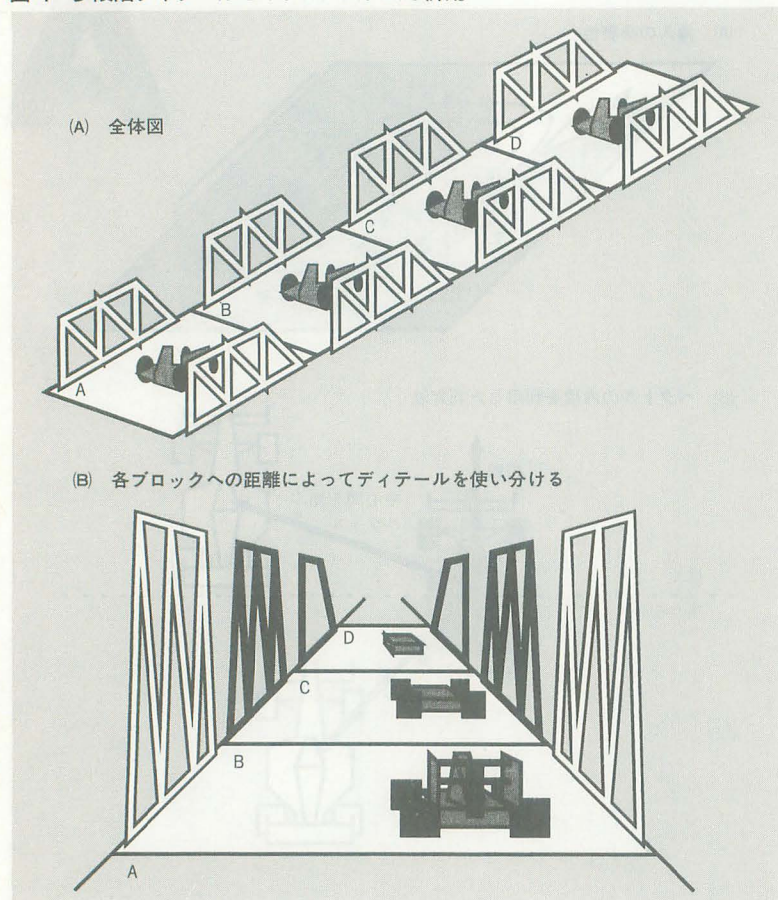



図5　ブロック間関係テーブル

(A)　正方向用テーブル

		to			
		A	B	C	D
from	A	3	3	2	1
	B	0	3	3	2
	C	0	0	3	3
	D	0	0	3	3

(B)逆方向用テーブル

		to			
		A	B	C	D
from	A	3	0	0	0
	B	3	3	0	0
	C	2	3	3	0
	D	1	2	3	3

1～3：そのディテールレベルで描画する
0　：描画しない

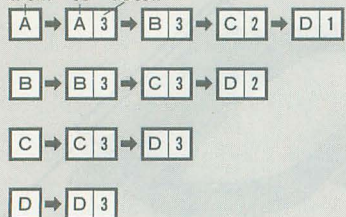
(C)　ブロック間関係リスト

from　to　detail

A→A 3→B 3→C 2→D 1

B→B 3→C 3→D 2

C→C 3→D 3

D→D 3

車の後方であり，まったく処理する必要がない。

ここで，図5(A)のようなブロック間関係テーブルを作ることで相当に楽ができる。ブロックがN個あると，テーブルの要素数はN×Nとなる。あるブロック（“from”ブロックとしよう）からあるブロック（“to”ブロックとしよう）はどう見えるか，どう処理すればいいかを数値化しておくのだ。テーブルの各要素は，“from”ブロックから“to”ブロックが見えるならばそのディテールのレベル（ディテールが3段階ならば1～3）を，見えないならば0の値をとる。

なお，このテーブルは，自分の車が正方向を向いている場合にしか使えない。サーキットを走る車なら，おおむね正方向を向いていると仮定してかまわない。ただ，3Dものの醍醐味はなんといっても逆走にあるという“通”なあなたや，いつまでたってもスピンばっかりしている未熟者のあなたのためなら，逆方向や横方向のテーブルも別に用意しておくのもやぶさかではない（図5(B)）。ということは，車がどの方向を向いているかということを判定するために，正方向のベクトルなるものを各ブロックごとに求めておく必要があるわけだ。

なお，より速度を稼げそうな戦略としては，あらかじめ各“from”ブロックごとに表示する“to”ブロックを求めておいてリストにしておくというものがある（図5(C)）。むろん，どのくらいのディテールで表示するかという情報も込みでリストにするのである。表示されないブロックを無駄にチェックすることがないので若干高速化できる。特にブロック数が膨大になったときに有利だ。この方式の欠点は，ほかの車がどの“to”ブロックに存在するかわかったとしても，表示するのかしないのか，どのくらいのディテールで表示するのかが即座にはわからないところである。

今後の予定

現在コースエディタを制作中である。ただ，SLASHver.1.0で書き始めたはいいのだが，ver.2.0向けの書き直しをまだ行っていない。現時点ではマップシステム風のものをCで書き，各ブロックのオイラー角を浮動小数点で計算しているので，結構遅い。とりあえずX68030でコプロセッサを直接駆動する（極悪なことに68882を載せたX68030でないと動作しない）ようにコンパイルオプションを設定しているが，それでもまだ遅いのだ。早々にSLASHver.2.0用のCライブラリを完成させねばならない。とりあえずコースデータをひとつ作り上げればいろいろと実験できるな。ということで能書きばかりの今回は終わる。それではまた次回。

SIDE B

SIDE B

リアリティのある映像とは?

Yokouchi Takeshi 横内 威至

**今回は，SLASHで使われている透視変換を取り上げる
そこには計算されたものを正確にモニタ上へ映し出すテクニックが必要だ
そして，より現実の映像に近づけるための手法を考えてみる**

先月は申しわけなかった。いきなりハードディスクがぶっ飛んで原稿が死んだため，やむなく連載を休んでしまうはめになるとは……。ハードディスクはしっかりバックアップをとること，飛びそうなソフトはディスクでしか使わないなどの政策転換をせねばならないときであろう。

さて，いよいよ氷点下を体感し，冬本場のアグレッシブな攻めの中，締め切りは俺をさらに追い込む。今月もまた座標系のあたりを解説していこう，という前にまた改造の投稿がきていた。今度は佐々木克博氏による平方根のアルゴリズムであった。ありがとう，さっそく採用させてもらった。そして，当然のように坪井氏も勝負に出ている。ガンガンとクロックアップにはげんでいるが,「ストリートファイターIIダッシュ」なんていう甘い割り込みなんかでしばらくは沈黙するのでは，との不安。誘惑に負けずさらなる協力をお願いする。

ついでにスピードを追求するのになぜグラフィックのポリゴンなのか，という質問があったのだが，ここでちょっと答えておこう。まずは初期バージョンということでグラフィックのほうが作りやすかったのが1つ。そして，もう1つ色数による表現力の問題。タイリングをサポートしても16色で表現するポリゴンはヘボイと思ったのである。スピードは当然追求すべき最大の要素であるが，リアリティもそこそこほしいから，結局現在の形となっている。

また，物体が小さければテキストもグラフィックもスピードに差が出ないであろう，と考えていた。8ドット以下の描画であればテキストを使うのはかなり不利であり，現実にはそのような描画のほうが多い，と推測していたのである。たとえポリゴンが水平を基本に描画されるとはいっても，テキストではエッジ部分での処理がかなりのネックになる。さらにプレーンが複数枚あるため，描画が重なる領域では，一度クリアしたうえで再描画を行わなければならないことなども考慮に入れると，テキストがグラフィックよりもそこまで有利なものとはいえないであろう。

具体的にフライトシミュレータを作る場合を考えよう。地上はテキストで処理するのがベストだが，そのほかのオブジェクトはどうだろうか。大きく表示される状態はかなり特殊な状況であるはずで，かえってテキストのほうが分が悪いようにも思える。いまのところあまりスピードは保証できない次期バージョンのテキスト版で同じデータを扱ってみたところ，なんとグラフィックのほうが速かったという事実もある。それなら表現力も上がるグラフィックを使ったほうが得策であろう。またシステム制作当初の甘い読みでは，マッピングまでも考えていたのかもしれない。以上の理由により，とりあえずグラフィックをメインにしているのである。

しかし，テキスト版も用意してあるので安心してもらいたい。一応シェーディングもつけたままでサポートする。読者諸氏にはいずれまた，テキストの描画ルーチンあたりを貪ってもらいたいと願うかぎりである。

図1

Z：x＝d：X

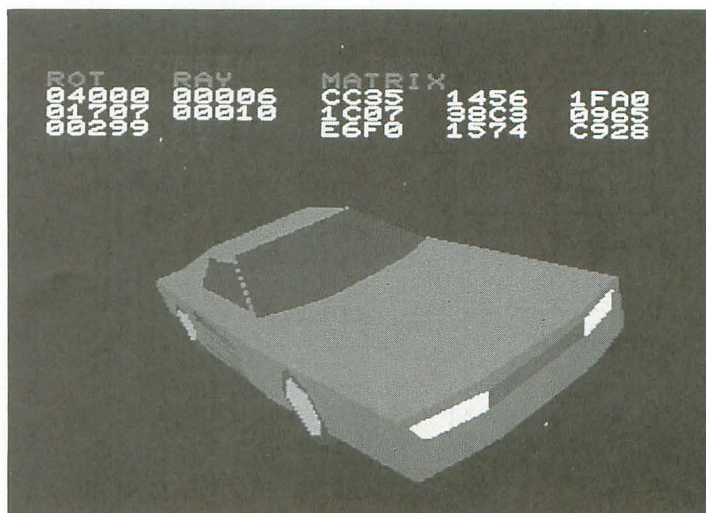
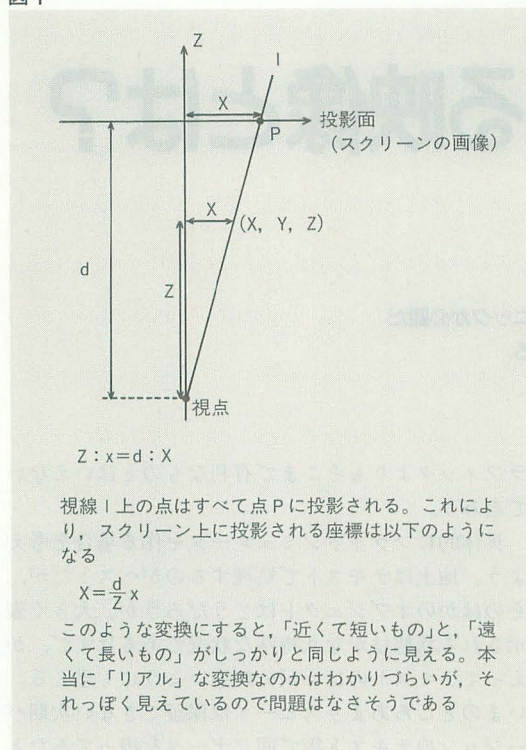
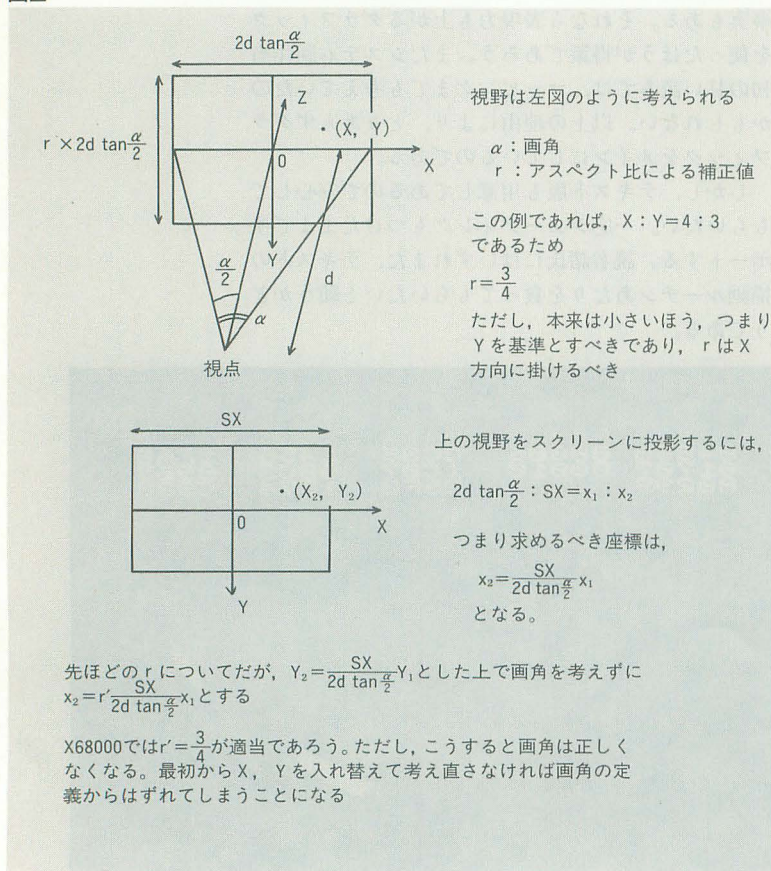
視線 l 上の点はすべて点Pに投影される。これにより，スクリーン上に投影される座標は以下のようになる

$$X=\frac{d}{Z}x$$

このような変換にすると，「近くて短いもの」と，「遠くて長いもの」がしっかりと同じように見える。本当に「リアル」な変換なのかはわかりづらいが，それっぽく見えているので問題はなさそうである

図2

先ほどのrについてだが，$Y_2=\frac{SX}{2d\tan\frac{\alpha}{2}}Y_1$とした上で画角を考えずに$x_2=r'\frac{SX}{2d\tan\frac{\alpha}{2}}x_1$とする

X68000では$r'=\frac{3}{4}$が適当であろう。ただし，こうすると画角は正しくなくなる。最初からX，Yを入れ替えて考え直さなければ画角の定義からはずれてしまうことになる

透視変換

「透視変換」。これは，極めて初歩的な領域である。とはいえ，何度もいっているように結構いやらしい部分をもっているため，しっかりと叩き込まねばならない領域である。

3Dの醍醐味といえばリアルな遠近感が挙げられる。当然，物体が遠くにあれば小さく見えるし，近ければ大きく見える。これは非常に感覚的なものであるため，適当な処理で実現するのがよいであろう。実際，ある物体がある距離にあった場合，どのようなサイズに見えるか，というのをモニタ上で計算することはナンセンスである。ここで，サイズ，といったがこれは長さの単位で表すことができない。そこで，画角で表すことになる。

では正確にフォーマットを決め，画角を正確に再現すればよいかもしれないが，モニタとの距離，そしてモニタのサイズで不確定となるため，画角も一切無視する。無駄な説明だったかもしれないが，まあとにかくそれらしく見せるだけだったら簡単な方法で解決できる問題だということである。

図1を見ていただきたい。これからスクリーン投影時に座標の拡大縮小のキーとなるのは，Z座標のみであることがわかる。ではここで出てきたdとはなんであろうか。これは次のステップで考えることにし，とりあえず現段階では「適当な補正値」ということにしておきたい。

基本的な透視変換というのは図1のような式で実現できるのだが，美しい画像を得るためにはさらに細工をしなければならない。不幸なことにドットアスペクト比を考慮しなければならないのだ。同時に，ここで画角を考慮することで望遠レンズのような効果をもたせることもできる。これは図2を見て考えてもらいたい。

本来ならばドットアスペクト比と画角，これで1セットと考えてもいいのだが，弱気なことにSLASHでは画角に対する処理は一切行っていない。無駄といえば無駄，ほかの処理でそこそこごまかせるものなので無視することにしていたためだ。しかし，あれば嬉しい効果なのでいずれ真剣に検討すべきかもしれない，とも考えている。

ドットアスペクト比の処理は簡単である。X68000ではドットが横方向に長いため，X方向を適当に縮小させるのがよいだろう。ドットアスペクト比がだいたい4：3だから，シフトで1/4倍した値を引くことでそれらしい画像になる。

最終的に，画面中央をデータで（0，0）とするために画面中心座標を加えるのを忘れないようにすること。

再びクリッピングについて

いままで，透視変換についていろいろ解説してきたが，このままではまずい状況が起こりそうである。Z座標が0以下のときには正しく処理できないのである。では，あらためてこの問題に取り組んでいこう。

まず点が何を意味しているかをはっきりさせたい。ポリゴン描画ライブラリであるため，これは当然ポリゴンの1点を示している。さらに，これはポリゴンを構成する辺の1点でもある。Z方向でクリッピングするには，この辺とクリッピング平面との関係を考えれば処理できることになっている。図3で軽く考えてみたい。

ではさらに，ポリゴンを処理することも含めて考えていこう。これは図4に示す。まず，ポリゴンを構成する点がすべてクリッピングにかかっていなければ普通に処理できる。そうでないときはクリッピングを行い，新しい座標を考慮したうえで処理しなければ，バックフェーシングさえもまともに処理できなくなってしまう。

まず，簡単に三角形で考えてみる。もし図4の例のように，頂点P_1がクリッピング平面より手前にきてしまったときにはどうなるだろうか。このとき頂点P_1を含む辺，つまり辺1，辺2とクリッピング平面との交点を新しいポリゴンの頂点として計算する必要がある。

この例でははみ出した頂点によって求めるべき点がわかっているため，処理はそれごとに分けてやればいい。これである程度は無駄を省けると予想できる。ただし，この処理がポリゴン単位で行われていれば，図4後半のような，同じ辺を2度計算している無駄が生じかねない。対策として，あらかじめ外れた点を使用するポリゴンを調べ，その点を通る辺をクリッピングした点すべてを新しく頂点データに加える，というのが考えられる。

しかし，これはかなり大掛かりな処理になるうえ，まともに実行したからといって，それに見合った速度を稼ぐにはいたらないし，座標変換の領域でポリゴンの都合が入ってくるのはあまりエレガントではない。現SLASHでは同じ計算を2度確実に行っているのはわかっているが，これはもう無視している。どちらにしても重い処理になってしまうからあまり考えてはいない。

迫力のある視点のために

俺はまだ「リッジレーサー」を見たことがないから「バーチャレーシング」のことを思い出してほし

図3

$X=\frac{d}{Z}x$ だが，$Z<0$ である

これはこのまま計算すると，P_2と同じ点に投影されてしまう

よって，ある一定の領域で点を無視しなくてはならない。理論的には$Z\leqq 0$でカットすべきだが，dとのバランスも考えるべきである

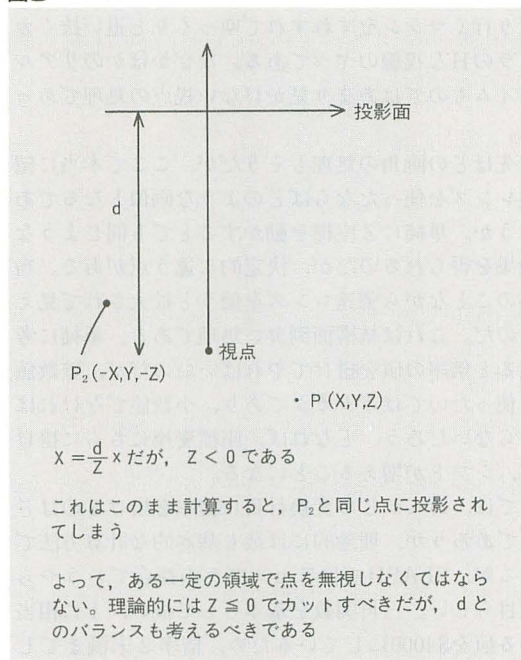
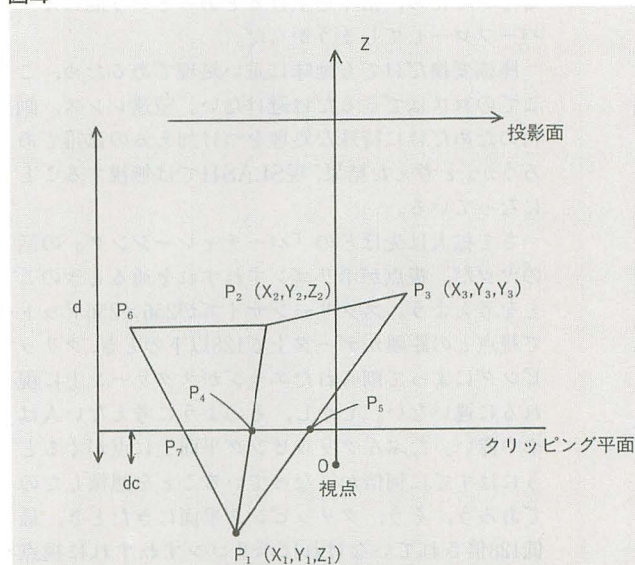



図4

辺P_1P_2とクリッピング平面の交点P_4 (X_4, Y_4, Z_4)

$$P_4=\overrightarrow{OP_1}+\frac{dc-Z_1}{Z_2-Z_1}\overrightarrow{P_1P_2}$$

辺P_1P_3とクリッピング平面の交点P_5 (X_5, Y_5, Z_5)

$$P_5=\overrightarrow{OP_1}+\frac{dc-Z_1}{Z_3-Z_1}\overrightarrow{P_1P_3}$$

以上のように点を作り，三角形$P_1P_2P_3$は四角形$P_2P_3P_5P_4$として扱う

別に三角形$P_1P_2P_6$があっても，やはり同じように四角形$P_6P_2P_4P_7$として扱うことになる，しかし，新しく作られた点P_4，P_5，P_7はその場限りの処理であり，この場合点P_4は2回計算されることになる。しかし，これも複雑な処理であるため，無視するのも手であろう

い。このゲームではデモがなぜか印象に残っている。走り行くマシンをすれすれでゆっくりと追い抜くカメラのHな視線のヤツである。なぜかほかのリアルタイムものではあまり見かけない視点の処理であった。

先ほどの画角の処理もそうだが，ここで本当に望遠レンズを使ったならばどのような画像となるであろうか。単純にZ座標を動かすことでも同じような効果を得られるのだが，決定的に違う点がある。当然のことながら望遠レンズを使うと拡大されて見えるのだ。これは結構面倒臭い処理である。単純に考えると倍率の値を掛けてやればいいのだが，整数値を使ったのではガタガタであり，小数値でなければならないだろう。となれば，座標変換にさらに掛け算，シフトが増えることになる。

では，あらかじめ変換行列に倍率を掛けるのはどうであろうか。理論的には最も基本的な計算方法であるが，SLASHではアセンブラの都合でどうやら駄目らしい。三角関数を基本として扱い，1に相当する値を$4000にしているため，倍率2未満までしか許せないことになっている。$8000は－1に相当しているため，倍率を上げるとあっという間にオーバーフローしてしまうからだ。

座標変換だけでも地味に重い処理であるため，ここでのロスはできるだけ避けたい。望遠レンズ，画角のためだけに特殊な処理をつけ加えるのは得であろうか。と考えた結果，現SLASHでは無視することになっている。

さて拡大は先ほどの「バーチャレーシング」の話のヤツだ。視点がポリゴンすれすれを通るときのことを考えよう。スクリーンサイズが256×256ドットで視点との距離がデータ上で128以下のとき，クリッピングによって削られたエッジがスクリーン上に現れるに違いない。しかし，そのように考えない人はやや偉い。たぶんクリッピング平面上に点がくるときにはすでに何倍かになっていることを想像したのであろう。そう，クリッピング平面にきたとき，最低128倍されていなければポリゴンすれすれに視点を置くことができないのである。そのための方法を考えよう。

ここで透視投影の計算を思い出してもらいたい。「適当な補正値」であった図1のdはこの領域を決定する重大な役割をもっているのである。例としてdを256とすると計算にはどのような影響が出てくるであろうか。図1の式を見ればわかるように，距離256のときにデータがそのままスクリーンでのサイズになる。クリッピング平面をZ＝16とすると，クリッピング時に16倍された画像が得られることになる。当然，美しくない副作用が出てくるのも忘れてはならない。

ということは，例のようにクリッピング時に16倍されているとなると，データ上で1の移動は16ドットもの移動になってしまうのである。ではそれを防ぐためにはどうすればいいかというと，単純にクリッピング時に1倍となるようにするしかない。また，同じことをしようとするなら，物体を非常に大きく作成することによっても解決できる。しかしながら，視点はポリゴンから離すことを前提としている。必然的に物体が大きくなれば物体の存在できる座標が相対的に狭められてしまい，空間が窮屈になるであろう。

どちらの方法を取るかはかなり微妙な問題であることがわかってくる。ここで，どちらにしても座標値がワードでしか表せないことがかなりのネックであることがうかがえる。恨むべきは掛け算，割り算が16ビットを基準としていること，ロングワード演算の低速度なのである。たぶんX68030専用版などでは，仕様も変えることでかなりまともになるのではないだろうか，と常々感じる次第である。

もうひとつ，これまで無視し続けた変換があるのだが皆はもう気づいていただろうか。画角のあたりでも考えなければならないことであるが，視野をモニタに映し出すときのことをもっと考えなければならない。もしも，画角を広げていったらどのような画像が得られるであろうか。本来ならばいずれは魚眼レンズのように変態的な画像になるはずである。

しかし，いままでのような変換では決してそのような画像は得られない。どこでこの差が出てしまうのかは考えればなんとなくわかってくる。視野は実際に1点を中心とし，その周りとの関係を表せるものは角度だけなのである。ところがモニタではそうはいかない。角度の代わりにドット数，つまりは距離でしか表すことができないのである。これは致命的であり，それなりの制限をつけないかぎり，正しい画像は作れない。見たところあまり不自然に感じないかもしれないが，極端な例を図5にて示す（ただし図5も嘘臭いかもしれないが)。

つまり，モニタだけでなくどんな絵でもそうだが，視野と距離をうまく合わせなければならないのだ。たとえばモニタに画角45度での表示をうまく行ったならば，ちょうど表示領域の端が45度に見えるような位置で眺めなければ，リアリティが得られないはずなのである。ではそのような条件をつけ，それなりの変換を行えばリアルに見えるのだろうか。おそらくそうはいかないと思う。慣れのせいもあるだろうが，あくまでもいままでのような「平面投射」のほうが理解しやすい画像となるだろう。これはこれで完成された手法なのである。

モニタが平面であり，しかもサイズがバラバラで空間上に存在する1つのオブジェクトとしてしか捉

らえられない以上は，平面投射が正しいのかもしれない。いつの時代か，また立体視がもてはやされたらしっかりと考え直してみてもよいのではないだろうか。右目，左目専用の２つの球形スクリーンを用意し，そしてドットに角度と１対１の関係をもたせればしっかりとした3D映像ができるのではないだろうか。ほとんど我流で，根拠のない考えだからあまり自信はないのだが。

制御することを学ぶ

基本的な，かつダークな部分についてはこのへんでとりあえず終わらせておきたい。まだ半端なままではあるが，基本とすべきことはだいたい説明したと思う。結果的にはSLASHで使われている方法が多くなってしまったが，まあそれはそれということで納得してもらいたい。

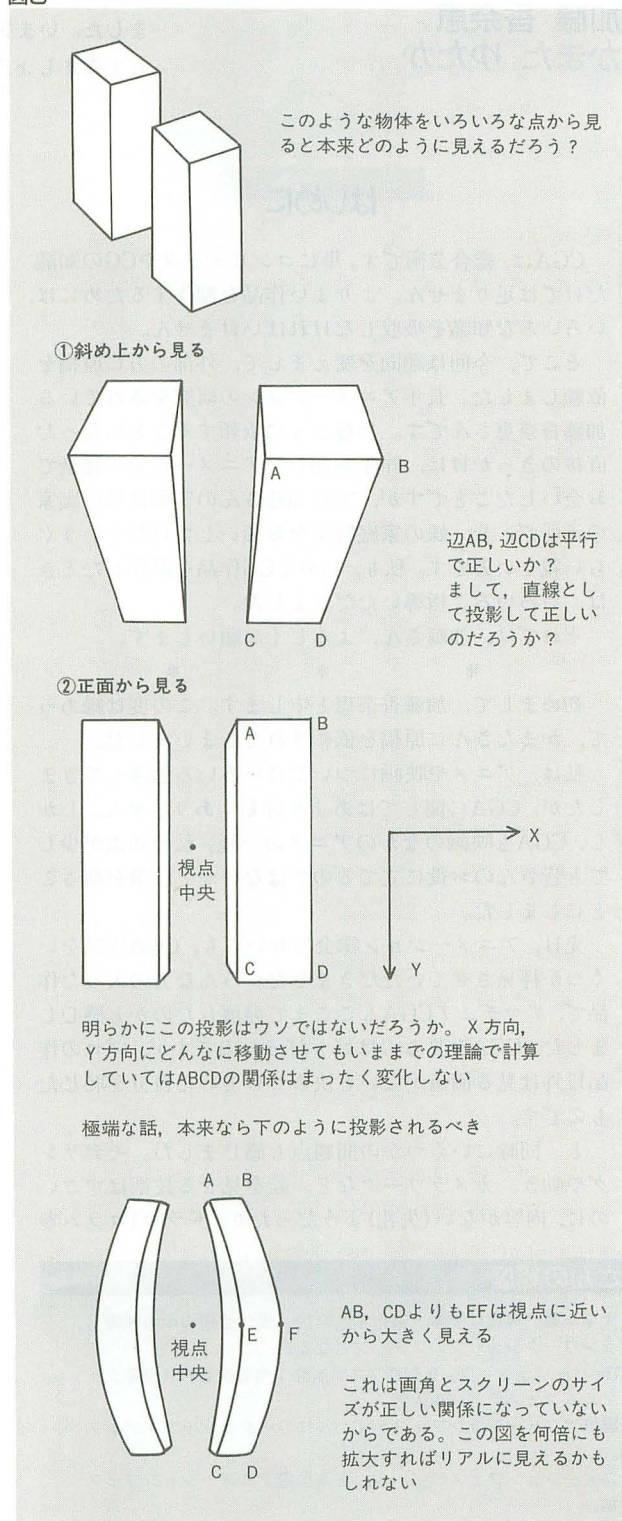
時間というか，そのほかいろいろな都合がつけば世に見受けられるいろいろな3D技術を考えてみたかったが，現実はシビアなのである。なにしろ先立って考えるべきことが多すぎる……というのは泣き言か。趣味の領域だが，レイトレースぐらい一度はやってみたかったのだが，これもあと回しということにしよう。

次回からは応用に入っていきたいと思っている。いよいよSLASHもバージョンアップし，やっとアセンブラを使ったサンプルを公表できるので，そのサンプルを研究課題として発展させていきたいと考えている。

まずは空間を支配するためのマップシステムを考えなければならない。しかし，ここでも座標系が鍵となっているので，このあたりの応用を固めるのが先だろうか。この２つの領域がしっかり見極められたならば簡単なサンプルとして遊び程度のものを作ってみたいと思っている。フライトシミュレータとまではいかないが，実際に飛び回れるサンプルを非常に作ってみたいのだ。シミュレータとなると，しっかりとした航空力学を考えなければならないので別問題だからだ。

さらにドライビングシミュレータとなると，もっといやらしい自動車力学が絡んでしまうので簡単には到達できないことになっている。シミュレータとなれば当然数ページにリストを納めることはできない。ましてアセンブラとなったらもう不可能。とりあえずは次回のサンプルリストでやっと次のステップの基本が公表できるのだが，それらの発展程度がいまのところ課題である。別件として困っていることはエディタである。マップステムと同時に作成しなくてはならないのだが，3Dエディタというのは非常にデザインしにくいものなので無視してしまっていた。これもまたやや苦しい作業が必要になってきている。

そういうことで，次回からはシステム内部事情を抜けてSLASHを制御することに移る予定だ。

図5

おしえて，アニメのえらい人

プロジェクトチームDōGA
加藤 香奈恵
かまた ゆたか

今回は，アニメーションに造詣の深い加藤さんに特別に原稿をお願いしました。いまアマチュアCGAに足りないものは何か，あらためて考え直してみましょう。

DōGA

はじめに

CGAは，総合芸術です。単にコンピュータやCGの知識だけでは足りません。よりよい作品を制作するためには，いろいろな知識を吸収しなければいけません。

そこで，今回は趣向を変えまして，外部の方に原稿を依頼しました。長年アニメーションの研究をされている加藤香奈恵さんです。加藤さんに依頼することになった直接のきっかけは，昨年参加したアニメーション総会でお会いしたことですが，実は加藤さんの実家は私の実家の近所で，昔，妹の家庭教師をお願いしていたというぐらい親しい方です。私も，初めてCG作品を制作したときは，いろいろご指導いただきました。

それでは，加藤さん，よろしくお願いします。

*　　*　　*

初めまして，加藤香奈恵と申します。この度は縁あって，かまたさんに原稿を依頼されてしまいました。

私は，アニメや映画についてはいろいろ勉強してきましたが，CGAに関してはあまり詳しくありません。しかし，CGAも映画のなかのアニメの一種。私の知識が少しでも皆さんのお役に立てるのではないかと，筆を執ることにしました。

先日，アニメーション総会においても，CGA作品をいくつか拝見させていただきました。みんな力の入った作品で，アマチュアCGAもここまで発展したのかと感心しました。CGAを見るのは好きだったのですが，プロの作品以外は見る価値がないと決めつけていた自分を恥じたものです。

と，同時にいくつかの問題点も感じました。モデリングや動き，カメラワークなど，絵を見せる技術はすごいのに，内容がない（失礼）ようだったり，ドラマ（コラム参照）作りの基礎が欠けていたりするからです。これはあまりにももったいないことです。

今回の話で，映画の作り方の基本的なことをわかっていただけたら幸いです。

加藤 香奈恵さんの経歴

- 1982年，同志社大学卒業（専攻は美学・芸術学。卒論「現代芸術の中の映画」）。
- 大阪アニメーションワークショップのスタッフとなる。
- 自主制作映画「Theパッパラショウ」を制作（共同制作：中谷美智枝（少年ナイフ））。オレンジフィルムフェスティバル入賞。
- 第1～3回広島国際アニメーションフェスティバルにワークショップ・アシスタントとして参加。
- ザグレブ国際アニメーションフェスティバル，上海国際アニメーションフェスティバルなどに参加。
- 1992年，小説「川は涙を飲み込んで…」で振姫文学賞2席を受賞。

ステップ1

作品作りの基本の基本。基礎の基礎。忘れられがちな部分。だけど，知っておかなければならないことです。

［1］動機

動機とは，「こんなCGAを作りたい」という自分の目標のことです。何をするときにも動機があるはずです。CGAを作るにも，殺人をするにも。いやいや，CGAも殺人も勢いでやってしまったという人が案外多いのかもしれませんが，それで困るのは警察だけではありません。あなた自身が困るのです。制作に行き詰まったとき，何でお金にもならないこんなことに時間を割いているのか悩んだとき，私はこのためにやっている，という確かな動機が支えになります。

過去のCGA作品にも，動機が明確な作品はいくつかあります。文月涼さんの『TORNADO』（X68000芸術祭グランプリ）は「自分のデザインした車を走らせたい」というのが動機です。三ッ木淳さんの『MACHINE VISUALIZATION』（第5回CGAコンテスト佳作）は「複雑な機械を視覚化したい，複雑な機械の仕組みを考える楽しさを知ってもらいたい」，そして昨年のCGAコンテストグランプリ作品，森山知己さんの『SWORD』は「現時点でのアマチュアCGAのもてる技術の最高を目指す」というのが動機でしょう。

これらの作品は，はっきりとした動機があったればこそ，多くの困難を克服する力を得て作品を完成させることができたのです。単に作者の力量がすばらしいというだけの問題ではありません。動機がなかったら，生まれなかった作品です。とはいっても本人に自覚はなかったかもしれません。動機とは陰の権力者なのです。

それではどうすれば動機が見つかるのでしょう。自分の好きなことや興味あることに注目してください。その近辺にきっと動機が見つかるでしょう。

メカのデザインに興味のある人は「自分のメカをどんなふうに活躍させたいのか」考えてみるといいでしょう。「美少女と巨大ロボットを闘わせたい」「宇宙船を太陽系を越えて飛ばせたい」「究極のドッグファイトを見せる

ぞ」などと広がっていくでしょう。

風景に興味のある人は，あるひとつの情景を切りとって考えてみてもいいでしょう。「彼女と見たオーストラリアの景色を再現したい」「朝露に映る雲を描きたい」など。

また，ほかにも「好きな曲に絵（CGA）をつけたい」「CGAを使って自分のアートのセンスを全開する」などというのもよいでしょう。

「CGA史上に残るスペシャルギャグを飛ばす」なんていう絵になるかどうかわからないものでも，動機としてはOKです。ただ，「X68000とDōGAシステムが手に入ったもんで……」なんていうのは動機とはいえません。

とにかく，いま自分が何をやりたいのかをよく考えるところからスタートしてください。

［2］テーマ（主題）

テーマとは，世の中（友達，恋人など）に向かって自分がいいたいことです。「え？　動機とテーマは違うの？」と叫んだ人はいませんでしたか。よく思い出してください。動機はやりたいこと。テーマはいいたいこと。微妙に違っているでしょう。この違いが大切なのです。

ただ，動機とテーマが直接関連している作品の場合，その区別は少し曖昧になります。『TORNADO』は「自分のデザインした車を走らせたい」という動機と「ねえ，この車すてきでしょう。みんな，見てくださいよ」というテーマがうまく嚙み合って成功した例でしょう。『MACHINE VISUALIZATION』も「複雑な機械を視覚化したい」という動機と「ほら，こんなにわかりやすいでしょう」というテーマがうまく作用し合っています。

しかし，ドラマ性の強い作品の場合，このように動機とテーマが直接関連しているのはあまり好ましくありません。また，多くの人に受けたいのならば，多くの観客が共感するようなテーマが必要です。ということはできるだけ普遍的なもので，人の心情に即したものがよいでしょう。

映画『バック・トゥー・ザ・フューチャー』は，その娯楽性だけが評価されがちですが，主人公マーティーの心の成長を描いた作品です。「自分の人生を自分の手で変える」というテーマは，観客に「自分もできるかもしれない」という希望を与えます。

また，テーマがしっかりしている映画は，それほど高い制作費をかけたものでなくてもヒットしています。『ゴースト』がそのよい例でしょう。「死んでもなお愛し合う恋人同士」というテーマが，人々の感動を呼びました。

テーマが必要だというと，とってつけたようなテーマを作る人がいますが，あくまで自分がいいたいことをテーマにしてください。京大マイコンクラブの『MOUSE』（第５回CGAコンテスト努力賞）には，最後にテーマらしきことをいうセリフがありますが，ああいうのをとってつけたテーマといいます。しかしながら，この作品は本当のテーマを別に隠しもっていたのです。「隣の芝生は青い。だからといって，横取りしては罰が当たる」という立派なテーマを。

なんで作者でもないのに隠されたテーマがわかるんだとお思いでしょう。それは，テーマは動機と違い，作品に顕著に表れる性質のものだからです。動機は作品に表れなくてもかまいませんが，テーマは必ず作品に表れなければなりません。「この作品が何をいいたいのか」が。逆にいうと，観客にわからないようなものはテーマではありません。正直な話，プロの作品でもテーマがはっきりしていないものは多くあります。でも，悪いところを見習うようなことはやめましょう。

では，どうしたらテーマが見つかるのでしょうか。たとえば，いいたいことを書き出してみるのもよいでしょう。難しく考える必要はありません。しかし，「おいしいものが食べたい」「彼女が欲しい」なんてのは，やりたいことですからテーマになりません。「近頃の外食はまずい。冷凍食品ばかりだからだ」「女性は男心を知らなすぎる」「世の中やっぱり金が一番」「でも，人情を忘れたらつらいよな」「強い者が勝つばかりじゃ，やりきれない」「趣味にすべてを賭けるって生き方もすてきだよな」など，身近なところから，テーマは生まれます。

自分がいいたいことが何も見つからない場合は，作品を作るのをやめたほうがいいと思います。そんな惚けた精神状態で，いいものが作れるわけがありません。

とにかく，いま自分が何をいいたいか，自分に問いかけてください。

［3］時間

自分の作品がおおよそ何分になるかは，最初に考えておきましょう。初めて挑戦する人が５分も10分もある作品を作ろうとすると，必ず挫折します。力量のある人が５秒しか作らないと，手抜きと思われます。力のある人はストーリー性のある作品に挑戦してください。

「応募するコンテストは時間が決められているのですが……」という場合もあるでしょう。しかし，普通，決められているのは最長の時間であって，短いぶんには問題ないと思います。自分の技量と制作時間で判断してください。

ステップ2

さて，初めはただロボットを闘わせていれば満足していた人も，次第にドラマのある作品が作りたくなります。しかし，ドラマ作りにはちゃんとした手法があるので，それを知らないと行き詰まってしまうでしょう。

また，ストーリーとキャラクターはどちらを先に作ってもかまいません。しかし，相互で矛盾が出ないように気をつけましょう。

［4］ストーリー

映画用語では，ストーリー，シノプシス，プロット（コラム参照），と呼び分けます。けれど，全部ひっくるめてストーリーと呼んで問題ありません。

まず，いわゆる５Wは必ず入れておきましょう。WHO，WHEN，WHERE，WHAT，WHY（誰が，いつ，どこで，何をした，なぜか）ということが必要です。時代や場所の設定はここで作っておきましょう。

ストーリーは時間進行順に作ります。

「一郎は紫煙の行方を追いながら思い出していた。あの日のことを……」

なんて，小説ふうに書く必要はありません。そうしなければストーリーが書けない人はそれでもかまいませんが。普通はストーリーといえば，

「1994年，コードネーム68000ダークスナイパーと呼ばれるやまだいちろうは，地球征服を狙って幼稚園バスの襲撃を続けるウハウハ団と闘っていた。一郎のもともとの仕事は，幼稚園バスの運転手だったが，彼の有給休日にバスがウハウハ団に襲われ，愛する幼稚園児たちが誘拐されたのだった」

といった，面白くもおかしくも美しくもない文章で十分です。ストーリーがそのまま作品になるわけではないのですから。

ストーリー作りは，テーマを念頭に入れておくことが必要です。先にストーリーがある場合も，主人公がここでなぜこのような行動したのかという部分にテーマを忍び込ませます。たとえば先のストーリーのテーマが「初恋が人生のすべて」だった場合，「やまだいちろうが勤める幼稚園には，初恋の女性の子供が通っていて，その子がウハウハ団にさらわれたのだ」ということにしてもいいでしょう。

[5]キャラクター

目に見えるデザインも大切ですが，面倒でも，あなたの作品に登場するキャラクターには性格や過去を与えてあげましょう。主人公の過去にテーマが隠されていれば，テーマを導きやすいでしょう。テーマが「必ず正義は勝つ」なら，「幼い頃，父が正義のための闘いに敗れて死んだ。しかし，主人公も父と同じ道を歩んでいた。今度こそ，正義が勝つと信じて」とするように。

余力がある人は，通りがかりの犬に至るまで，ちゃんと設定してください。「その犬が秘密の鍵を握っていた」などがあるとストーリー展開がスムーズにいきます。逆にいうと，意味もなく何かを登場させるのはやめたほうがよいということです。

たとえ自然を描く作品であっても，花や木にも設定は必要です。「紅葉＝15歳。両親の顔は知らないけれど，そんなことで落ち込んだりはしない。ちょっとおちゃめで力持ちの中学生。楓君に片思い中」なんてのは冗談ですが，風や雨に，あるいはどんな風に反応するかなどは設定しておきましょう。「風で引きちぎられるように，葉っぱが飛ぶ木」とか，「強風では，枝はしならないで，ベキッと折れてしまう」といった感じです。

メカでも同じです。戦闘機ひとつでも，どんな攻撃に弱いのか，武器は何か，どんな作戦に有効か，ちゃんと考えておきましょう。浅野英史さんの『MISSION』(第5回CGAコンテストアクション賞）などの作品では，登場するメカがそれぞれの特性をもっています。ただこの作品では，ストーリーがないのが寂しいのですが。

ドラマを盛り上げるための設定の一種として，「枷」というのがあります。ウルトラマンが3分間しか闘えないとか，変身しないと普通の人だとか，合体しなければ最大のパワーが出ないとか，一回使うとエネルギー充電に時間がかかる武器，などといった「制約」を作るのです。

[6]絵コンテ

シナリオ（コラム参照）の代わりに絵コンテを作りましょう。もちろんシナリオを作ってからでもかまいません。しかし，アマチュアCGA作家は，シナリオの書き方に悩む時間があるのならば，それよりも作業を先に進めたほうがいいでしょう。ここではシナリオの書き方は省きます。必要ならば自分で勉強してください。『アニメ・シナリオ入門』(鳥海尽三著）がお薦めです。

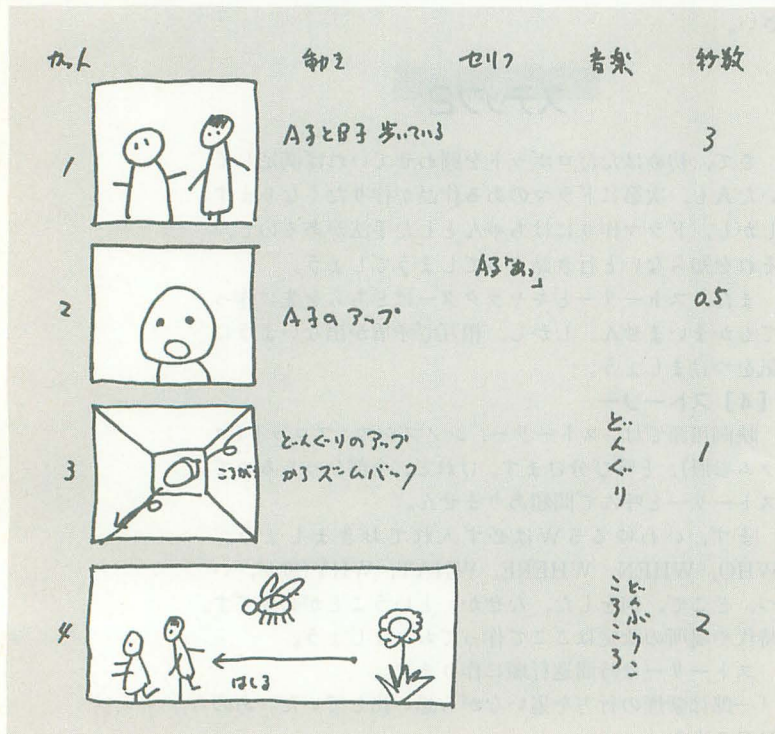
絵コンテには，作品に必要なすべてのカットを描き出し，そのときのせりふまたは状況説明，タイムを入れます。カメラの動きなども，メモしておくといいでしょう。使う曲が決まっている場合，歌詞や楽譜を入れておきましょう。しかし，ここに示した例は，私のやり方です。これがベストというわけでは決してありません。要は，使うべき絵が客観的に見られるようにすることです。

「西暦2300年，超能力をもつ新人類と，機械の体をもつ超人類と，変化のなかった人類とが対立していた。超人類は宇宙ステーションに居住していたが，そこは耐久年数の限界がきていた。云々」などというナレーションやせりふを冒頭にもってくるのはお勧めできません。

映画『スター・ウォーズ』ではやってるじゃないかという人がいると思いますが，あれは，タブーに挑戦した一種の遊びです。事実，あのテロップがなかったとしても内容がわからなくなるものではなかったでしょう。

せっかく思い通り（にいかない場合も多々ありますが）の絵が作れるCGAなのですから，絵に語らせましょう。状況を説明するのに簡単な方法としては，オープニングで音楽にのせて，というのがあります。カットを多く入れたほうが，内容を深く語れるでしょう。音楽は効果的に使えば大きな武器になります。

ひとつの作品に何カット必要かという目安は特にありませんが，5分の作品で10カットしかないというのは，あまりにも少なすぎます。長く見せる技術をもっている

図1 絵コンテの例

人でも難しいでしょう。

CGAの場合，1分間に15カットくらいが見やすいと思います。逆にアクションシーンでも，40カットは超えないほうがよいでしょう。目がチカチカしてしまいます。『MISSION』はカットの長さのメリハリが効いているので，40カット近くあっても見やすくなっています。

［7］演出

まずは，絵コンテの修正から入りましょう。第1稿の絵コンテは，映画のシナリオに当たります。監督が制作する段階でシナリオを修正するのと同様に，絵コンテの問題点をここで直しておきます。［6］の絵コンテまでは感性，センス，ノリがものをいいました。でも，ここは冷静な目が必要な知的作業です。じっくり考えてみましょう。

チエックすべきことは無数にあるので，今回は最も重要なことだけにしておきます。

1）起承転結がありますか。起とは物語の導入部。観客に「おっ」と思わせて見る気にさせなければいけません。承とは観客に「どうなるのかな」と興味を持続させて後半へともっていく部分です。伏線などはここで入れておくといいでしょう。転とは，観客に「うっそー！」とか「行けー！」と夢中にさせるクライマックスの部分です。テーマがここで前面に出てきます。結とはテーマを観客に納得させ，余韻をもたせる部分です。

宍戸光太郎さんの『A PLANET』（第5回CGAコンテスト佳作）には，見事に起承転結があります。起は星へ落ちていくところ。承は男がさまよい歩き，腕を切られる。転は村を見つけ，駆けていって胴を切られる。結は飢え死を待つ。というふうに，解説できるくらい明確です。拍手。

2）絵コンテに書かれたタイムは，予定時間と大幅に違っていませんか。違っているとすれば，カットが足りないか，多いか，または1カットの時間が正しくないか，です。

タイムが均等割りになっていませんか。各カットが全部3秒ずつなんてのは見苦しいですね。

3）モンタージュ（コラム参照）がちゃんとできていますか。たとえば，驚く人のアップの直後のカットには，その人が驚いた原因の絵をもってきましょう。驚く顔の次に雲をもってきたりすると，雲に驚いたことになってしまいます。カットはそれぞれ前後で関連しあって成り立っています。

4）テーマはちゃんと表現できていますか。どうすればテーマが明確になるかは，はっきりいって明快なアドバイス方法がないのですが，たとえば『バック・トゥー・ザ・フューチャー』で主人公のマーティーが「ぼくが人生を変えてみせる」と話すように，最初に主人公にテーマを語らせるという方法があります。また，主人公の友人や敵方にテーマと相反する行動をさせるというのもあります。

5）ストーリーそのままを絵にしていませんか。映像はストーリーを省略して見せるところが魅力です。「昔，仲のよかったAとBが，ある理由で戦争を始めた」という話でも，戦いの部分から始めていいのです。

以上，自分で気づいたことはチェックして，問題がある部分は順序を入れ替えるか，あるいは削ってください。CGAに限らず，一度作ってしまった映像を削るのは，不可能といってもいいくらい，辛いものです。作る前によく検討してください。

ステップ3

では，実際に制作に入りましょう。このあたりの話は，何度もかまたさんがされていると思います。もう，私などがあまり出る幕のないところです。

［8］制作

CGAの制作自体は，皆さんのほうが詳しいでしょうから省略します。あなたのもてる力を注いで作画に励んでください。

［9］編集

作画して，それで終わりではありません。せりふを入れたり，音楽を入れたり，という作業ももちろんあります。しかし，それも完了したあと，もうちょっとつきあってください。

CGAの場合，編集という言葉が適当かどうかわかりませんが，つまりは最終確認です。完璧に計画していても，できあがってみるとなんだか違う，ということもあります。できた作品を見て，カットのつながりが悪かったり，

▶CGAコンテスト事務局より◀

ついに，第6回アマチュアCGAコンテスト発表会が開催される。みんな3月6日（日）は空けてあるな。場所と時間は右記の通りだから，もう一度ちゃんとチェックしておくこと。会場が去年と違うぞ。間違って新宿や銀座に行くんじゃないぞ。

さて，今年のエントリー作品を紹介したい……ところなのだが，残念ながらそうもいかない。実は，一次審査（選外だけを決める）がまだ終わっていないのだ。なぜ，今年はこんなに審査が遅れているかといえば，そう，予想通り応募総数が多いのだ。ええこっちゃ。

だから，どれが入選するか決まっていないうちから解説するのは危険だが，パッと見た感じでは，なんかいままでと傾向が違うなという印象がある。ひとつには，アート系の作品が多いようだ。それと，去年以上に新人が多い（ただし選外も多そう）。また，よいことか悪いことかわからないが，CGAシステムの割合が減っている。あと，バトルロボットものが元気だ。バトルロボットだけで10作品以上あるんじゃないかな。

今年の発表会は，例年以上にイベント（内容は秘密）が多くなるそうなので，「ビデオを申し込むからいいや」なんていわずにぜひご来場ください。

＊　＊　＊

第6回アマチュアCGAコンテスト
入選作品発表会（入場無料）

●東京地区
日時：1994年3月6日（日）
PM1：00（開場），PM1：30（開演）～PM5：00
場所：三宅坂ホール（社会文化会館内）
地下鉄有楽町線「永田町」（出口2）下車3分

●大阪地区
日時：1994年4月2日（土）
PM1：00（開演）～PM4：00
場所：摂津市民文化会館
JR京都線「千里丘」下車　南へ徒歩15分

動きが内容に合っていない，特に，音楽と絵の動きが不釣り合いになっている場合は，勇気を出して直してください。アマチュアの作品の多くは，明らかに時間切れ，という作品が目立ちます。締め切りがある場合は，特に余裕をもって作品を作ってください。

[10] 人目にさらす

たったひとりで作った場合も，グループで作った場合も，身近な人に作品を見せることは大切です。人には好みがありますから，全員が絶賛しなくても，ひとりでも作品を理解してくれる人があれば，次回作の励みになります。コンテストに送るだけでは，入選でもしない限り批評は聞けません。つらくても，批評を聞くことも大切な修行です。制作者は不屈の精神を養わなくてはいけません。

[復習] 不敵な行為！ 『SWORD2』を切る

『SWORD 2』を見たときは，こんなことまでできるのか，と驚きました。でも，これを見て，この原稿を書くことを決心したともいえます。

今回の復習として，『SWORD 2』を例に挙げて，その問題点などについて検討したいと思います。『SWORD 2』を俎上に載せるのは，昨年のグランプリ受賞作品であるというだけでなく，アマチュアCGA作家にありがちな問題点を含んでいるからです。それと，かまたさんが「森山さんなら，怒るような人じゃないから安心して書いていいよ」とおっしゃっていたからです。

1）動機

先ほどの動機の項でも述べたように，この作品にははっきりとした動機があります。皆さんも見習ってください。

2）テーマ

残念ながら，この作品には明確なテーマがありません。バーチャルリアリティに関して何かいいたかったような感じもありますが，私にはよくわかりませんでした。

内容から考えると，「ただの男でも勇気を出せば，勇者の剣と盾，美女だって得ることができる」というテーマにすれば作りやすかったでしょう。

3）時間

『SWORD 2』は，約8分の作品ですが，適当な長さといえます。これだけの長さの作品ができる人は，そんなに多くないでしょう。

長ければいいというわけではありませんが，力の入った作品があまりにも短いと不満が残ってしまいます。

4）ストーリー

テーマが明確でなかったため，ストーリーが作りにく

夫婦でQ&A

うさ子：おかげさまで，無事タイから戻ってまいりました。

ゆたか：いやー，タイのお寺はキンキラキンでしたね。もう，壮絶ですよ。

うさ子：私は，「暁の寺」が気に入りました。高い塔に登るのがスリル満点でした。

ゆたか：ちゃんと，スタッフへのおみやげ“闇鍋PART2の具”も買えたし。

うさ子：本当に，あんなもの食べたの？

ゆたか：いや，結構おいしかったそうやで。私は怖くて参加しなかったけど。

うさ子：でも，あれって……本当に食べ物だったのかしら。

ゆたか：スタッフのみんなにはそういっといたけど……。この話は内緒にしておきましょうか。

* * *

Oさん(葛飾)：**かまたさん，東京に怪獣が現れることはもうないです。ツブラヤが手を引いたらしい。**

ゆたか：まだ，東映が……。

うさ子：ツブラヤって，フグ料理の？

ゆたか：あんたは，ひとりでボケてなさい。

Sさん(八王子)：**どうしても教えてもらいたいことがあるのです。X68030Compactを買ったのですが，どうやってもCGAシステムVer. 2.50（1992年7月号の付録）を起動させることができません。**(以下，症状報告続く)

ゆたか：この件は，以前掲載したと思うのですが，相変わらず毎月のようにお問い合わせがあります。Ver.2.50は，X68030では起動できません。いちばんてっとり早い解決方法は，当方に最新バージョンを送るよう依頼していただくことです。

うさ子：1992年7月には，X68030はできてませんでしたから，対応はできなかったのです。当たり前ですね。

Tさん(世田谷区)：**かまたさん，へんなマンガ描いてごめんね。**

ゆたか：T先生，CGAコンテストの審査，よろしくお願いします。こんな私でよければ，なんなりとネタにしてください。

うさ子：や～い，目がハート。

ゆたか：責任の半分は，おまえだろ。

Tさん(埼玉)：**PIXEL読みました。レイトレの性能を生かしたシステムが発表されるって本当ですか？ だったらすごく楽しみです。頑張ってください。**

ゆたか：ウソです。

うさ子：えっ，ウソ書いたんですか？

ゆたか：いや，見込みはあったんや。反射，屈折なんかはかなり高速になりそうなんやけど，影落ちが遅くなって，結局メリットなかってん。いや～，残念。なんなら，CGAシステムの形状データやモーションデータがそのまま使える，むっちゃくちゃ遅いレイトレを発表しましょうか？

うさ子：むちゃくちゃ遅いって，どのくらい遅いのですか？ ……えっ！ そんなに！ よっぽどヒマな人しか使えないじゃない。

Tさん(荻窪)：**誰か封印してくれー。残り3カ月だー！**(11月受理のお便り)

うさ子：受験ご苦労さまです。もうそろそろ3カ月になりますが，どうでしたか。

ゆたか：当チームでは，受験生向けに，一定期間X68000を預かり，その間，皆さまに代わって有効に活用するというサービスを始めます。ご利用ください。

Nさん(香川)：**「今年はバリバリ作品作るでーと意気込んでいたのですが，時間がなーい！で，代わりに曲を作っていきたいなと思っています。そこで，質問です。最近はやりのSMFデータでの応募は，ダメなのですか？**

ゆたか：SMFって何？

うさ子：スタンダードMIDIファイルです。問題ないと思いますので，ぜひ，よろしくお願いします。

T(川口北)：**うさ子愛してるよ（会ったことないけど)。**

ゆたか：ゆるさん！ おまえなんか「さん」つけたらへん！

Iさん(焼津)：**CGAマガジンの芸術祭オープニングCGを，Macintoshユーザーに見せたらバカにされた。そして，Macの雑誌の付録のCD-ROMに収められていた『SWORD 2』を見せられて，「どーだ，Macは凄いだろう」……。**

ゆたか：Macの雑誌とは，『MacUser』のことですね。実は，我々の担当だったOh!X編集部のAさんが，そちらの編集部に移られたので，友情出演ということでCGAコンテストの作品を提供したんです（Quicktimeに変換)。

うさ子：みんな自分のマシンに思い入れがあるんですね。

ところで，話を「闇鍋」に戻すけど，本当におなか壊した人いないの？

ゆたか：大丈夫だって，日頃から「3秒ルール」で鍛えてるから。

うさ子：なんやの，その「3秒ルール」って？

ゆたか：床に落ちた食べ物でも，3秒以内なら，拾って食べても大丈夫という，衛生的になんの根拠もないルール。

うさ子：ゆたかさんも，鍛えておいてね。

ゆたか：おまえ，どんな料理食わしてんねん？

DōGA

かったのでしょう。私が勝手につけた上記のテーマだと，「1993年，不毛の地の迷宮で，勇者の剣と盾を取り戻すべく魔物と闘う元王女。気楽な旅を楽しんでいた男は，ふらっと立ち寄った廃墟に，魔物に捕らわれた美女（元王女）を見てしまう。彼女の美しさに惹かれて迷宮に足を踏み入れた男は，魔物に襲われる。初めは逃げているだけだったが，次第に，自分が彼女を助けるという意識が芽生える。魔物を倒し，勇者の剣を得て，美女を助けるべく闘う」というようなストーリーが出てきます。

5）キャラクター

テーマ，ストーリーが弱いために，キャラクター作りもやりにくかったのでしょう。全体の登場人物の設定は一応できていますが，主要キャラクターの設定が足りません。

女はなぜそこにいたのか。

戦闘力が強いのか，弱いのか。

男はなぜそこへ来たのか。

なぜ魔物と闘いはじめたのか。

翼竜（怪鳥）はどういう役割があるのか。

など，もう少し詳しい設定が必要です。

6）絵コンテ

ちゃんとできていたのでしょう。カメラワークやモンタージュの技法は高度なものが取り入れてあります。あれだけテーマやストーリーが弱いのに，これだけ見せてしまう力量は見事です。

7）演出

演出上の問題はほとんどないのですが，よくできているがゆえ，気になることを挙げておきます。

前半部分で鳥が見ている視点のカメラワークが多用されていました。きっと，翼竜の視点だと思います。それにしては，翼竜の活躍がありません。カメラは普通，作者の目か，観客の目か，主人公の目か，物語上重要な登場人物（動物）の目を表現するものです。登場人物の目としてよくあるのは，犯人の目というパターンです。『SWORD 2』において，翼竜の視点を使うならば，もっと翼竜に重要な役割を担うような行動をさせるべきです。廃墟の主だとか，実は味方だったとか。ほかの魔物と同レベルではいけません。

また，男が主人公だと思いますが，登場が遅すぎます。2時間の映画で，1時間経って主人公がやっと出てくるようなものです。主人公はちらっとでもいいから，初めに登場させましょう。砂漠に小さく見える赤い車，というように。

8）制作

本文同様，省略します。

9）編集

『SWORD 2』ぐらいのレベルになると，「このカット，あと，0.3秒短いほうがいいのに」というような細かな問題が出てきます。しかし，それは作者の好みなどもあるので，本人が満足していたら直さなくてもいいでしょう。

10）人目にさらす

グランプリを受賞するような作品でも，人目にさらすことで，今回のようにいろいろいちゃもんをつけられるのです。制作者には，強い精神力が必要だというのもよくわかったでしょう。

＊　　＊　　＊

『SWORD 2』のいちばんの欠点はやはり，「テーマがない」ということにつきると思います。テーマがないと構成がしにくいという典型的な例です。

ということで，以上いろいろ話してきましたが，要は，どんなことをするにも基礎が大切だということです。確かに，基礎の勉強ってつまらないものですけれど，避けて通るわけにはいきません。CGA制作に当たって，この講座のことを，頭のすみっこにでもとどめてもらえるとうれしいです。

あれこれ勝手なことを書きましたが，例に使われた作品の作者の方々，ごめんなさい。基本的に，嫌いな作品は取り上げていません。これも，愛するがゆえの鞭だと理解して許してください。あまりお役に立てなかったかもしれませんが，これから，すばらしいCGA作品がたくさんできることを願って筆を置きます。

専門用語の解説

ドラマ

作られた物語。感動，感傷を伴っている。

モンタージュ

単一のカットではなく，複数のカットを組み合わせることによって，1つのカットでは表現できないようなイメージを表現する方法。

シナリオ

シーン別に分かれ，ト書きとせりふによって成り立つ文章。映画の台本。脚本と同じ。

ストーリー

ストーリー<プロット<シノプシス，この順で場面設定，状況設定，人物設定が詳しくなる。いわゆる，映画のあらすじのこと。

おわりに

以上，加藤さんのお話でした。いままで，結構いい加減な気持ちでCGAを作っていた私としては，ずいぶんと思い当たるフシがあったりして……。皆さんは，いかがでした？

なお，今回掲載したのは，あくまで加藤さんのご意見であって，DōGAとして，「テーマのない作品は認めない」とかいうつもりはありません。

しかしながら，現在のアマチュアCGAは，年々レベルが向上しているとはいえ，作品性とか内容という点では，まだまだほかの映像メディアに劣る面があります。そのあたりの問題をクリアすることで，大きな発展が期待できると思います。今年のCGAコンテストは，いかがでしょうか？　楽しみですね。

＊　　＊　　＊

さて，今後のことですが，半年前からある計画を練っていたんですが，コンテストもあるし，まだ，その計画は準備が進んでいません。ということで，次回はお休みで5月号よりまた心機一転，連載を再開します。

それではまた，お楽しみに。

Oh!X LIVE in '94

©NAMCO ALL RIGHTS RESERVED「Winning Run」より

X68000・Z-MUSIC用
(CM-64対応)

THEME FROM WINNING RUN

Fukui Yuki
福井 祐貴

©TECMO「スターフォース」より

X68000・Z-MUSIC用
(SC-55対応)

スターフォースアレンジ版

Shoji Shingo 荘司 真吾

春は目前。人間やらパソコンやら封印していた人も，そろそろ活動再開かな。新しい始まりの季節の2曲は，元気よくゲームミュージックといきましょう。ますますハイレベルで突っ走るLIVE inですが，初心者の方の乱入もお待ちしていますよ。

エントリー受け付け中

今月は久しぶりにゲームミュージックオンリーでいきましょう。1曲目はナムコの3Dレーシングゲームです。といっても，話題の「リッジレーサー」ではありませんので，あしからず。その「リッジレーサー」の先祖(?)ともいえる「Winning Run」から「THEME FROM WINNING RUN」をお届けしましょう。演奏にはCM-64が必要です。

「Winning Run」は，当時としては斬新なソリッドモデルを使用した，シミュレーションと呼べるような驚異的な3Dレーシングゲームでした。慣れるまではまっすぐ走るのすら難しいという感じで，F1クラスの車のすごさをいやというほど味あわせてくれました。「リッジレーサー」からテクスチャマッピングをとった感じですね。

作品のほうはといえば，歌うクレジット音(コインを入れたときのSE)と呼ばれる「Feel the beat of the Winning Run」というフレーズはかなりイメージをつかんでいますね。全体的にみれば，原曲ではもっとファンキーな音色が多数使用されていますよね。そのせいか，ちょっとおとなしく聴こえてしまいます。もうすこし歯切れのいい音を使ってもよかったかもしれませんね。もちろん，掲載されるレベルなのですから，このままで十分楽しめる仕上がりにはなっています。最後はループにせずにうまく処理すれば，もっとよかったかもしれません。

ところで，「リッジレーサー」のCDが発売されてからそろそろ1カ月。誰かこちらのほうも挑戦する人はいませんかねぇ。

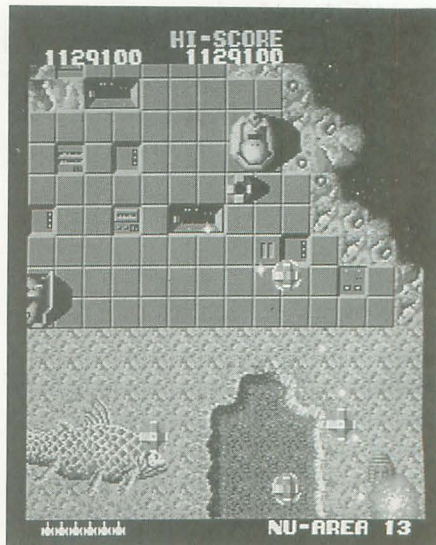
Winning Run

星軍参上！

さて，2曲目はX68000用に電波新聞社から発売されているビデオゲーム・アンソロジー・シリーズの3作目「スターフォース」より，「スターフォース アレンジ版」をお届けしましょう。こちらの作品はSC-55同等品が必要です。

懐かしの「スターフォース」ですが，このゲームをやって熱くなった人は多いハズ。元が10年前という時代背景もあり，シンプルなシューティングゲームなのですが，昨年の春から夏にかけて，指を痙攣させていた人もいたんじゃないかな。

この作品の題材は刺激的なことこのうえなしで，さらにアレンジをぶちかましているのです。アレンジといっても，そんじょそこらのアレンジではありません。作者の荘司君については「サンバDEグワッシャ!!」(1993年2月号)といえば思い出す人も多いでしょう。あの強烈なイメージは忘れられるもんじゃありません。

今回はかなりゴージャスにまとめられています。LIVEを意識してるのかな？ ソロもばっちり入れてあるし，かなりうまく仕上がってますね。前作「サンバDEグワッシャ!!」に比べると，原曲の雰囲気がずいぶんと残っていますが，それだけにテーカン(現テクモ)によるアレンジかと思ってしまうほどです。そろそろ常連の仲間入りをしそうな荘司君ですが，常連の人々と比べても，アレンジャーとしてのレベルはかなり高い部類に入りますね。これからも「腕っこき」目指して，精進してくださいね。みんなもこれに刺激されて，「忠実移殖」だけでなく自分なりのアレンジに挑戦してみると面白いかもよ。 (SIVA)

スターフォース

リスト1 THEME FROM WINNING RUN

```
   1: .COMMENT  WINNING RUN  "THEME FROM WINNING RUN" (C)NAMCO
Programed & Arranged by FUKUCHANG
   2:
   3: /::::::::::::::::::::::::::::::::::::::::::::::::::::::::
   4: /:                                                      :
   5: /:             [[  WINNING RUN (C)NAMCO  ]]             :
   6: /:                                                      :
   7: /:             [  THEME FROM WINNING RUN  ]             :
   8: /:                                                      :
   9: /:                 Programed & Arranged by YUUKI FUKUI  :
  10: /:                                       FOR  ZMUSIC.X  :
  11: /:                                  MIDI MODULE  CM-64  :
  12: /:                                                      :
  13: /::::::::::::::::::::::::::::::::::::::::::::::::::::::::
  14:
  15: (I)
  16:
  17: / TRACK SETUP  ::::::::::::::::::::::::::::::::::::::::::
  18:
  19: / CM-64  ::::::::::::::::::::::::::::::::::::::::::::::::
  20:
  21: (M17,4000)(AMIDI 2,17)
  22: (M18,4000)(AMIDI 3,18)
  23: (M19,4000)(AMIDI 4,19)
  24: (M20,4000)(AMIDI 5,20)
  25: (M21,4000)(AMIDI 6,21)
  26: (M22,4000)(AMIDI 7,22)
  27: (M23,4000)(AMIDI 8,23)
  28: (M24,4000)(AMIDI 9,24)
  29: (M25,8000)(AMIDI10,25)
  30: (M26,4000)(AMIDI11,26)
  31: (M27,4000)(AMIDI12,27)
  32: (M28,4000)(AMIDI13,28)
  33: (M29,4000)(AMIDI14,29)
  34: (M30,4000)(AMIDI15,30)
  35: (M31,4000)(AMIDI16,31)
  36:
  37: / CM-64 INIT  :::::::::::::::::::::::::::::::::::::::::::
  38:
  39: .ROLAND_EXCLUSIVE 16,22={$7F,00,00,00}
  40:
  41: / CM-64 SYSTEM SETUP  :::::::::::::::::::::::::::::::::::
  42:
  43: / LA SOUND PART  ::::::::::::::::::::::::::::::::::::::::
  44:
  45: .ROLAND_EXCLUSIVE 16,22 ={
  46:           $10,0,0                 /ADDRESS
  47:           64                      /MASTER TUNE
  48:           1,3,5                   /REVERB
  49:           4,4,4,4,4,4,4,0,4       /PTL RESERVE
  50:           1,2,3,4,5,6,7,8,9}      /MIDI CH#
  51:
  52: / PCM SOUND PART  :::::::::::::::::::::::::::::::::::::::
  53:
  54: .ROLAND_EXCLUSIVE 16,22 ={
  55:           $52,0,0                 /ADDRESS
  56:           64                      /MASTER TUNE
  57:           1,3,5                   /REVERB
  58:           7,7,7,7,0,0             /PTL RESERVE
  59:           10,11,12,13,14,15}      /MIDI CH#
  60:
  61: / MML DATA SET  :::::::::::::::::::::::::::::::::::::::::
  62:
  63: /      [  CHOIRUS & STRINGS  ]  :::::::::::::::::::::::::
  64:
  65: (T26) T119
  66: (T26) R2
  67: (T26) @33O4L2@V110@P64@K0@U127
  68: (T26) 'A<C+E>''B<EG+>'&'B1<E1G+1>'
  69: (T26) [SEGNO]
  70: (T26) @35O4L16@V88@P94@K2@U127
  71: (T26) |:3R1:|
  72: (T26) R2RAG+EA8G+F+&F+1&F+2RAG+F+A8G+E&E1&E2
  73: (T26) RAG+EA8G+F+&F+1&F+2RAG+EB8G+E&E1&E2
  74: (T26) RAG+EA8G+F+&F+1&F+2RAG+EA8G+E&E2.RB<DE&E>B<DE&E>B'
<DF+>''<EG+>'R2
  75: (T26) [TOCODA]
  76: (T26) |:6R1:|
  77: (T26) 'DF+''C+E''>B<D''>G+E<'R8Q4'D8F+8'R8'D8F+8'Q8'DF+'
'C+E''>B<D''>G+E<'
  78: (T26) 'DF+''C+E''>B<D''>G+E<'R8Q4'E8G+8''>A8.<D8.F+8.''>
A+8.<D+8.G8.''>B8<E8G+8'Q8
  79: (T26) [D.S.]
  80: (T26) [CODA]
  81: (T26) |:8R1:|
  82: (T26) [DO]
  83: (T26) |:4R1:|
  84: (T26) [LOOP]
  85:
  86: (T27) R2
  87: (T27) @35O4L2@V64@P64@K0@U125
  88: (T27) 'A<C+E>''B<EG+>'&'B1<E1G+1>'
  89: (T27) [SEGNO]
  90: (T27) @35O4L16@V88@P34@K-2@U127
  91: (T27) |:3R1:|
  92: (T27) R2RAG+EA8G+F+&F+1&F+2RAG+F+A8G+E&E1&E2
  93: (T27) RAG+EA8G+F+&F+1&F+2RAG+EB8G+E&E1&E2
  94: (T27) RAG+EA8G+F+&F+1&F+2RAG+EA8G+E&E2.RB<DE&E>B<DE&E>B'
<DF+>''<EG+>'R2
  95: (T27) [TOCODA]
  96: (T27) |:6R1:|
  97: (T27) 'DF+''C+E''>B<D''>G+E<'R8Q4'D8F+8'R8'D8F+8'Q8'DF+'
'C+E''>B<D''>G+E<'
  98: (T27) 'DF+''C+E''>B<D''>G+E<'R8Q4'E8G+8''>A8.<D8.F+8.''>
A+8.<D+8.G8.''>B8<E8G+8'Q8
  99: (T27) [D.S.]
 100: (T27) [CODA]
 101: (T27) |:8R1:|
 102: (T27) [DO]
 103: (T27) |:4R1:|
 104: (T27) [LOOP]
 105:
 106: (T28) R2
 107: (T28) @31O3@V78@U120
 108: (T28) (O2A4.<F),24&(F2B),0&@W1&@W8
 109: (T28) [SEGNO]
 110: (T28) @31O4L8@V68@P34
 111: (T28) |:4R1:|
 112: (T28) F+1&F+2F+.F+16&F+G+&G+1&G+2A.G+16&G+F+&F+1&F+2A.A1
6&AG+&G+1&G+2
 113: (T28) A.G+16&G+F+&F+1&F+2A.A16&AG+&G+1&G+2R2
 114: (T28) [TOCODA]
 115: (T28) |:8R1:|
 116: (T28) [D.S.]
 117: (T28) [CODA]
 118: (T28) |:8R1:|
 119: (T28) [DO]
 120: (T28) |:4R1:|
 121: (T28) [LOOP]
 122:
 123: (T29) R2
 124: (T29) @31O3@V78@U120
 125: (T29) (C+4.B),24(B2<E),0&@W1&@W8
 126: (T29) [SEGNO]
 127: (T29) @31O4L8@V68@P64
 128: (T29) |:4R1:|
 129: (T29) D1&D2D.D16&DE&E1&E2F+.E16&ED&D1&D2F+.F+16&F+E&E1&E
2
 130: (T29) F+.E16&ED&D1&D2F+.F+16&F+E&E1&E2R2
 131: (T29) [TOCODA]
 132: (T29) |:8R1:|
 133: (T29) [D.S.]
 134: (T29) [CODA]
 135: (T29) |:8R1:|
 136: (T29) [DO]
 137: (T29) |:4R1:|
 138: (T29) [LOOP]
 139:
 140: (T30) R2
 141: (T30) @31O3@V78@U120
 142: (T30) (E4.<D),24&(D2G+),0&@W1&@W8
 143: (T30) [SEGNO]
 144: (T30) @31O4L8@V68@P94
 145: (T30) |:4R1:|
 146: (T30) >A1&A2A.A16&AB&B1&B2<D.>B16&BA&A1&A2<D.D16&D>B&B1&
B2<
 147: (T30) D.>B16&BA&A1&A2<D.D16&D>B&B1&B2<R2
 148: (T30) [TOCODA]
 149: (T30) |:8R1:|
 150: (T30) [D.S.]
 151: (T30) [CODA]
 152: (T30) |:8R1:|
 153: (T30) [DO]
 154: (T30) |:4R1:|
 155: (T30) [LOOP]
 156:
 157: /      [  TRUMPET 1  ]  :::::::::::::::::::::::::::::::::
 158:
 159: (T17) R2
 160: (T17) |:2R1:|
 161: (T17) [SEGNO]
 162: (T17) |:4R1:|
 163: (T17) @89O4L8@V119@P64@K0@U127
 164: (T17) R16<A16DQ4G+EF+>Q8B16<E16R4R16D16DQ4C+C+Q8>B16R.R4
 165: (T17) R16<G+16EQ4F+DEQ8>B16<D16R4R16>B16BQ4AAQ8G+16R.R4
 166: (T17) 16<A16DQ4G+EF+>Q8B16<E16R4R16D16DQ4C+C+Q8>B16R.R4
 167: (T17) R16<B16EQ4ADQ8G+E16F+16R4R16D16DQ4C+C+>Q8B16R.R4
 168: (T17) R16<A16DQ4G+EQ8F+>B16<E16R4R16D16DQ4C+C+Q8>B16R.R4
 169: (T17) R16<G+16EQ4F+DQ8E>B16<D16R4R16>B16BAA16G+16R2
 170: (T17) [TOCODA]
 171: (T17) |:8R1:|
 172: (T17) [D.S.]
 173: (T17) [CODA]
 174: (T17) |:8R1:|
 175: (T17) [DO]
 176: (T17) |:R1:|
 177: (T17) |:<C+16D16C+16>B16R16<Q4E>Q8B16<C+16D16C+16>B16R4:
|
 178: (T17) [LOOP]
 179:
 180: /      [  TRUMPET 2  ]  :::::::::::::::::::::::::::::::::
 181:
 182: (T18) R2
 183: (T18) |:R1:|R32.
 184: (T18) [SEGNO]
 185: (T18) |:4R1:|
 186: (T18) @89O4L8@V94@P64@K-2@U120
 187: (T18) R16<A16DQ4G+EF+>Q8B16<E16R4R16D16DQ4C+C+Q8>B16R8.R
4
 188: (T18) R16<G+16EQ4F+DEQ8>B16<D16R4R16>B16BQ4AAQ8G+16R.R4
 189: (T18) R16<A16DQ4G+EF+>Q8B16<E16R4R16D16DQ4C+C+Q8>B16R.R4
 190: (T18) R16<B16EQ4ADQ8G+E16F+16R4R16D16DQ4C+C+>Q8B16R8.R4
```

```
   191: (T18) R16<A16DQ4G+EQ8F+>B16<E16R4R16D16DQ4C+C+Q8>B16R.R4
   192: (T18) R16<G+16EQ4F+DQ8E>B16<D16R4R16>B16BAA16G+16R2
   193: (T18) [TOCODA]
   194: (T18) |:8R1:|
   195: (T18) [D.S.]
   196: (T18) [CODA]
   197: (T18) |:8R1:|
   198: (T18) [DO]
   199: (T18) |:R1:|
   200: (T18) |:<C+16D16C+16>B16R16<Q4E>Q8B16<C+16D16C+16>B16R4:
|
   201: (T18) [LOOP]
   202:
   203:
   204:
   205: /      [  ELECTORIC ORGAN  ]  :::::::::::::::::::::::::::
   206:
   207: (T19) R2
   208: (T19) |:R1:|
   209: (T19) [SEGNO]
   210: (T19) |:8R1:|
   211: (T19) @6O4L16@V96@P64
   212: (T19) R2'AB<D>'RR'AB<D>'R4
   213: (T19) |:3R1:|
   214: (T19) R2'AB<D>'RR'AB<D>'R4
   215: (T19) |:3R1:|
   216: (T19) [TOCODA]
   217: (T19) |:4R1:|
   218: (T19) @26O4L16@V70@P64@K0
   219: (T19) |:F+ED>G+<R8Q4F+8R8F+8Q8F+ED>G+<F+ED>G+<R8Q4G+8Q8F
+ED>G+&G+4<:|
   220: (T19) [D.S.]
   221: (T19) [CODA]
   222: (T19) |:8R1:|
   223: (T19) [DO]
   224: (T19) |:4R1:|
   225: (T19) [LOOP]
   226:
   227: /      [  SYNTH. BRASS  ]  ::::::::::::::::::::::::::::::
   228:
   229: (T20) R2
   230: (T20) |:2R1:|
   231: (T20) [SEGNO]
   232: (T20) @26O4L16@V69@P94@K0
   233: (T20) R4'A8.<D8.F+8.>'E'A<DF+>''B<EG+>'R'A<DF+>'R4
   234: (T20) R'A<DF+>'BEQ4'A8<D8F+8>''B8<E8G+8'Q8'A<DF+'R8.R4
   235: (T20) R4'A8.<D8.F+8.>'E'A<DF+>''B<EG+>'R'A<DF+>'R4
   236: (T20) R'A<DF+>'<E>DQ4'A8<D8F+8>''B8<E8G+8'Q8'A<DF+'RRQ4'
EB<D>'RQ8'F+8.B8.<D8.>'
   237: (T20) R'A8<D8F+8>'E16Q4'A8<D8F+8>''B8<E8G+8>'Q8'A<DF+>''
B<EG+>'R'A<DF+>'R4
   238: (T20) R'A<DF+>'R8Q4'A8<D8F+8>''B8<E8G+8>'Q8'A<DF+>'R8.'B
24<D24>'A24G+24A16G+16
   239: (T20) R4'A8.<D8.F+8.>'D'A<DF+>''B<EG+>'R'A<DF+>'R4
   240: (T20) R'A<DF+>'R8Q4'A8<D8F+8>''B8<E8G+8>'Q8'A<DF+>'RR'EB
<D>'&'EB<D>'R8.
   241: (T20) R'A<DF+>'R8'A<DF+>'R8.'A<DF+>''B<EG+>'R'A<DF+>'R4
   242: (T20) R'A<DF+>'R8Q4'A8<D8F+8>''B8<E8G+8>'Q8'<EA>''<EG+>'
R'<DF+>'R'<EG+>''<D8F+8>'
   243: (T20) R'A<DF+>'R8'A<DF+>'R8.'A<DF+>''B<EG+>'R'A<DF+>'R4
   244: (T20) R'A<DF+>'R8Q4'A8<D8F+8>''B8<E8G+8>'Q8'A<DF+>'RRQ4'
A<D>'Q8R8'A<D>''B<E>'
   245: (T20) R4'A8.<D8.F+8.>'E'A<DF+>''B<EG+>'R'A<DF+>'R4
   246: (T20) RQ4'A<DF+>'Q8BEQ4'A8<D8F+8>''B8<E8G+8>'Q8'A8<D8F+8
>'A+8'B8<D8>''A+8<C+8>'
   247: (T20) R4'A8.<D8.F+8.>'D'A<DF+>''B<EG+>'R'A<DF+>'RB<DE&E>
B<DE&E>B<DER2>
   248: (T20) [TOCODA]
   249: (T20) |:R1:|
   250: (T20) |:3DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE
&E4<:|
   251: (T20) [D.S.]
   252: (T20) [CODA]
   253: (T20) |:4R1:|
   254: (T20) |:2DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE
&E4<:|
   255: (T20) [DO]
   256: (T20) |:2DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE
&E4<:|
   257: (T20) [LOOP]
   258:
   259: (T21) R2
   260: (T21) |:2R1:|
   261: (T21) [SEGNO]
   262: (T21) @26O3L16@V69@P34@K0
   263: (T21) E4R4R4RB<D>E&E8R8R4R4RB<DE>
   264: (T21) E4R4R4R<DE>E&E8R8R4R8Q4<E8>R4Q8
   265: (T21) ER8.R4R4R<DD+E>ERR<D>R4R2
   266: (T21) E4R4R4RB<D>BE8R8R4R8Q4B8Q8RB<DE>
   267: (T21) E8<D>BR8Q4<D8>Q8R4RB<DE>E8.<DR8Q4E8Q8F+ERDRED8>
   268: (T21) E8<D>BR8Q4<D8>Q8R4RB<DE>ER8.R8<E8D8>B8&BB8.
   269: (T21) E4R4R4RB<DER2R8E8R>B<DE>E4R4R4R>B<DE&E>B<DE&E>B<DE
R2
   270: (T21) [TOCODA]
   271: (T21) |:4DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE
&E4<:|
   272: (T21) [D.S.]
   273: (T21) [CODA]
   274: (T21) |:4DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE
&E4<:|
   275: (T21) [DO]
   276: (T21) |:DC+>BE<R8Q4D8R8D8Q8DC+>BE<DC+>BE<R8Q4E8Q8DC+>BE&
E4<:|
   277: (T21) [LOOP]
   278:
   279: /      [  ACOUSTIC BASS  ]  :::::::::::::::::::::::::::::
   280:
   281: (T22) R2
   282: (T22) |:R1:|
   283: (T22) [SEGNO]
   284: (T22) |:3R1:|
   285: (T22) @65O2L16@V108@P64@K0
   286: (T22) R2R<ED>BA+G+F+8
   287: (T22) |:R8Q4E8Q8E4DE8ER8<C+D>REERQ4E8Q8EER2:|
   288: (T22) R8Q4E8Q8E4DE8ER8<C+D>REERQ4E8Q8EER8<C+>BR<C+>B8
   289: (T22) R8Q4E8Q8E4DE8ER8<C+D>REERQ4E8Q8EER8.<D&D>B<DE>
   290: (T22) R8Q4E8Q8E4DE8ER8<C+D>REERQ4E8Q8EER2
   291: (T22) R8Q4E8Q8E4DE8ER>B<DE&E>B<DE&E>B<DER2
   292: (T22) [TOCODA]
   293: (T22) |:4<DC+>BER8<Q4D8R8D8Q8DC+>BE<DC+>BER8Q4<E8Q8DC+>B
E&E4:|
   294: (T22) [D.S.]
   295: (T22) [CODA]
   296: (T22) |:4<DC+>BER8<Q4D8R8D8Q8DC+>BE<DC+>BER8Q4<E8Q8DC+>B
E&E4:|
   297: (T22) [DO]
   298: (T22) |:<DC+>BER8<Q4D8R8D8Q8DC+>BE<DC+>BER8Q4<E8Q8DC+>BE
&E4:|
   299: (T22) [LOOP]
   300:
   301: /      [  DRUMS  ]  :::::::::::::::::::::::::::::::::::::
   302:
   303: (T25) R2
   304: (T25) O2L16@V127
   305: (T25) R1C24C24C24'D8C+8''DC+''D8C+8''DC+'R'DC+'R'DC+'<C2
4C24>A24A24G24G24
   306: (T25) [SEGNO]
   307: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   308: (T25) 'CF+'F+'CF+'F+'DC+'F+'CF+''CF+G+''F+G+''CF+''CF+'F
+'DC+A+'F+'F+A+'F+
   309: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   310: (T25) 'CF+'F+'CF+'F+'DC+'F+'DC+F+C''CF+'F+'FC+'F+F+'DC+'
'DC+''D8C+8'
   311: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   312: (T25) 'CF+'F+'CF+'F+'DC+'F+'CF+''CF+G+''F+G+''CF+''CF+'F
+'D8C+8''D8C+8'
   313: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   314: (T25) 'CF+'F+'CF+'F+'DC+'F+'CF+A''CF+G+''F+G+''CF+''CF+'
F+'DC+A+'F+'F+A+'F+
   315: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   316: (T25) 'CF+'F+'CF+'F+'DG+'F+'CF+A''CF+G+''F+G+''CF+''CF+'
F+'C+A+'F+'F+A+'F+
   317: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   318: (T25) 'CF+'F+'CF+'F+'DC+'F+'CF+A''CF+G+''F+G+''CF+''CF+'
F+F+'DC+''DC+''DC+'
   319: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+'F+'DC+''
D+F+G+'F+'D+F+G+'
   320: (T25) 'CF+'F+'CF+'F+'DC+'F+'CF+''CF+G+''F+G+''CF+''CF+'F
+'DC+A+'F+'F+A+'F+
   321: (T25) 'CF+G+'F+F+F+'DC+'F+'CF+''CF+'F+'CF+''CF+''F+G+'R1
6'DC+''DC+''CF+G+'
   322: (T25) R'DC+''DC+''CF+G+'R'DC+''DC+''CF+G+'R4'DC+''DC+'C8
   323: (T25) [TOCODA]
   324: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   325: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+DA<G+>'F+F+F+
   326: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   327: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+8D8A8''C+8D8A8'
   328: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   329: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+DA'F+F+F+
   330: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   331: (T25) 'C+D''C+D''C+D''C+D''C+D''C+D''C+D''C+D''<C+8.>C8.
''C8.<C+8.>''C8<C+>'
   332: (T25) [D.S.]
   333: (T25) [CODA]
   334: (T25) |:'<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'
F+'C+8D8A8<G+8>'F+F+
   335: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+DA<G+>'F+F+F+
   336: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   337: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+8D8A8''C+8D8A8':|
   338: (T25) [DO]
   339: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   340: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+DA<G+>'F+F+F+
   341: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+''CF+'<G+>C'CF+'F+
'C+8D8A8<G+8>'F+F+
   342: (T25) '<C+8G+8>C8'F+F+'C+8A8<G+8>''CF+A+''CF+G+A+''F+<G+
>''CF+G+''CF+'F+'C+8D8A8''C+8D8A8'
   343: (T25) [LOOP]
   344:
   345: (P)
```

▶機種の選択に間違いはないと思った。　宝福　公司(25)北海道

リスト2　THEME FROM WINNING RUN用カウンタ表示

```
17:000025E0 00000300  18:000025E9 00000300  19:000025E0 00000300  20:000025E0 00000300
21:000025E0 00000300  22:000025E0 00000300  25:000025E0 00000300  26:000025E0 00000300
27:000025E0 00000300  28:000025E0 00000300  29:000025E0 00000300  30:000025E0 00000300
```

リスト3　スターフォース アレンジ版

```
  1: /*******************************************************/
  2: /*              H O T   H O T   S H O T ! !            */
  3: /*                from Star Force (C)TEKAN             */
  4: /*             programmed by しょうじしんご            */
  5: /*******************************************************/
  6:
  7: .comment - HOT HOT SHOT!! from Star Force((C)TECMO)
- by しょうじしんご
  8: / for SC-55(155) or CM-500(300)
  9:
 10: /-----------------------------------------------------
 11: / TRACK SETUP
 12:
 13: (i)
 14: (bl)                / Base Channel = MIDI
 15:
 16: (m10,3000)(aMidi10,10)
 17: (m11,3000)(aMidi10,11)
 18: (m12,3000)(aMidi10,12)
 19: (m13,3000)(aMidi10,13)
 20: (m1 ,3000)(aMidi1 , 1)
 21: (m2 ,3000)(aMidi2 , 2)
 22: (m3 ,3000)(aMidi3 , 3)
 23: (m4 ,3000)(aMidi4 , 4)
 24: (m5 ,5000)(aMidi5 , 5)
 25: (m6 ,3000)(aMidi6 , 6)
 26: (m7 ,3000)(aMidi8 , 7)
 27: (m8 ,3000)(aMidi7 , 8)
 28: /-----------------------------------------------------
 29: / SC55 INIT
 30:
 31: .roland_exclusive 16,66={$40,00,$7F,00}
 32:
 33: /-----------------------------------------------------
 34: /VOICE RESERVE
 35:
 36: /                      1  2  3  4   5  6  7  8
 37:
 38: .sc55_v_reserve $10={2, 2, 2, 2,  3, 4, 2, 2,
 39:
 40:                        0, 5, 0, 0,  0, 0, 0, 0}
 41:
 42: /                      9 10 11 12  13 14 15 16
 43:
 44: /-----------------------------------------------------
 45: / MML DATA SET
 46: (o182)
 47:
 48: (t1 ) @i$41,$10,$42 @e90,50
 49: (t2 ) @i$41,$10,$42 @e90,40
 50: (t3 ) @i$41,$10,$42@e120,30
 51: (t4 ) @i$41,$10,$42@e120,30
 52: (t5 ) @i$41,$10,$42@e120,20
 53: (t6 ) @i$41,$10,$42@e120,10
 54: (t7 ) @i$41,$10,$42@e120,50
 55: (t8 ) @i$41,$10,$42@e120,90
 56: (t10) @i$41,$10,$42 @e50,50
 57:
 58: (t1 )     @m   @q0@g12@b0@p 34@38@v110@u120 o1@k0r418
 59: (t2 )     @m   @q0@g12@b0@p 74@31@v100@u120 o2@k5r418
 60: (t3 ) @h20@m50 @q0@g12@b0@p 28@31@v100@u120 o3@k5r418
 61: (t4 ) @h40@m120@q0@g12@b0@p100@30@v95 @u120 o4@k0r418
 62: (t5 )     @m   @q0@g12@b0@p44 @ 3@v95 @u120 o5@k0r418
 63: (t6 )     @m   @q0@g12@b0@p10 @49@v90 @u120 o5@k0r8...18
 64: (t7 )     @m   @q0@g12@b0@p64     @v120@u100 o4@k0r414
 65: (t8 )     @m   @q0@g12@b0@p64     @v120@u120 o3@k0r418
 66: (t10) @r1 t182              @17@v90 @u120 o2   r418
 67: (t11) @r1                                  o3   r411
 68: (t12) @r1                                  o3   r4116
 69:
 70:
 71: (t1)    /Track 1 -------------------------------------
 72: rggggggg
 73:
 74: <|:c-cccc-cccrc-ccc-ccc|>efffefffreffefff<:|
 75: l4dc>bagfe
 76:
 77: <g4   cr2.r2>eg<cr2.r2cdl8|:d+d+d+d+d+d+d+d+:|
 78: |:3cccccccc:|>ggggr2
 79:
 80: r1r1_10|: (b-b+)&r4g4b-4 b+4b+b-b+grb-4:|
 81:       |:|:8(b-b+)&r4g4b-4|b+4b+b-b+grb-4:||b+4b+b-b+grb-4
 82:
 83: |:4g+g+gg+4g4g+4g+g+4f4gg+|a+a+aa+4a4a+4a+a+4a4a+4:|
 84: rg4rgg4gr2.b-4:|
 85:
 86: ~l5<c4ccc>l4|:10fe-crrc8c8rrce-frr|f8f8rr:|
 87: rl8b<ccce-e-ff
 88: |:>|:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-|
 89:   >|:ab-b-b-:|rab-b-ab-b-b-<l4c>b-ag<fe-dc:|
 90: l8>|:ga-a-a-:|<|:de-e-e-:|>|:ab-b-b-:|rfffffff
 91:
 92: |:|:ab-b-ab-b-ab-|<effeffef>:|<de-e-de-e-de->:|
 93: <|:|:effeffef:|||:de-e-de-e-de-:|:|de-e-e-efff>
 94: |:ab-b-b-:|rfffffff<
 95:
 96: /2つ上の段と似ています。
 97:   |:|:>|:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-
 98: >||:ab-b-b-:|rab-b-ab-b-b-<l4c>b-ag<fe-dc:||l8
 99:  >|:ga-a-a-:|<|:de-e-e-:|>|:ab-b-b-:|rfffffff:|_15
100: |:|:ga-a-a-:||rga-a-ga-a-a-:|r~l5fffffff
101:   |:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-
102: |:efff:|l4<c>b-agfe-dc>b-8r8a2fllb-*678@b0,-8192,35b-4
103:
104: (t2)    /Track 2(t1と似ています。)-----------------
105: rggggggg
106:
107: <|:(c-4c)cc(c-4c)ccr(c-4c)c(c-4c)cc
108:  | (e4f) ff(e4f) ffr(e4f) f(e4f) ff:|l4dc>ba<gfe
109:
110: >(g4c)cr2.r2<eg<cr2.r2cdl8|:d+d+d+d+d+d+d+d+:|
111: |:3cccccccc:|>ggggr2
112:
113: >r1r1_10|: b-b+rgrb-r b+4b+b-b+grb-r:|
114:        |:|:8b-b+rgrb-r|b+4b+b-b+grb-r:||b+rb+b-b+grb-r
115:
116: |:4g+g+gg+rgrg+rg+g+rfrgg+|a+a+aa+rara+ra+a+rara+r:|
117: rgrrggrgr2.b-r:|<
118:
119: ~l5c4ccc>l4|:10fe-crrc8c8rrce-frr|f8f8rr:|rl8b<ccce-e-ff
120: |:>|:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-|
121:   >|:ab-b-b-:|rab-b-ab-b-b-<l4c>b-ag<fe-dc:|
122: l8>|:ga-a-a-:|<|:de-e-e-:|>|:ab-b-b-:|<rfffffff>
123:
124:  |:|:ab-b-ab-b-ab-|<effeffef>:|<de-e-de-e-de->:|
125: <|:|:effeffef:|||:de-e-de-e-de-:|:|de-e-e-efff>
126:  |:ab-b-b-:|rfffffff<
127:
128: /2つ上の段と似ています。
129: |:|:>|:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-|
130:      >|:ab-b-b-:|rab-b-ab-b-b-<l4c>b-ag<fe-dc:||
131:   l8>|:ga-a-a-:|<|:de-e-e-:|>|:ab-b-b-:|rfffffff:|_15
132:    |:|:ga-a-a-:||rga-a-ga-a-a-:|@64r>|:7f~8:|@31_41<
133:      |:ab-b-b-:|rab-b-ab-b-b-<|:de-e-e-:|rde-e-de-e-e-
134: |:efff:|l4<c>b-agfe-dc>b-8r8a2fllb-*678@b0,-8192,35b-4
135:
136: (t3)    /Track 3 -------------------------------------
137: rggggggg_5
138:
139: <|:'c2g''c4.g''cg'r'cg''cg'r'c2g'
140:  | 'f2a''f4.a''fa'r'fa''fa'r'f2a':|frerdrcr>brargrfr
141:
142: _5<ggg@b0,-180,0grgr@b0arg&r2.ggg@b0,-180,0grgr@b0a
143: ~5rg&r2.grggrgrgrg4.grfr(fe)&r2..&r*384r1
144:
145: l1rrrrrr
146: |:@z120,,,80,60,20o3|:rrrr:|l8de-e-e-4e-4&r1&r1&r2..
147: e-4f1@v90d1~5|f1~5g2_10<<gfe-d_15
148:
149: r8l4|:c1&r2>a-<e-d1&r2>b-<fe-1&r2|
150: cgf.e-8d.c8>b-8<c8d2.:|~15l8e-fgrgrrggrgr1:|
151:
152: >_5f2..~l5g4gl16fe-crl8|:10q8f4e-4>q4grb-b+r|:gb-b+|r:|
153: q8<c4e-4q4cre-fr|:ce-f|r:|:|_5
154:
155: q8l4e-f|:'df'&r1>b-<dq4fq8'e-2g'&r1
156: |(g8a)&r8{gr}'d1f'&r2dfc1(ef)&rgf:|
157: e-ga-1g2e-g'f1b-''f2a''fg'@b0,-8192,0~30<<'fa'@b0_15
158:
159: o3l8|:b-rrb-rrb-r(ga)rrargarb-rrb-rrb-r<(de-)rrdrr|cr>:|
160: <e-r(e-f)rrfrrfr(e-f)rre-rrdr(de-)rre-rre-re-rrdrre-
161: r(e-f)rrfrrfrgrrarrb-r_10<crrcr(cd)cr>b-r1l4a.ga
162:
163: /2つ上の段と似ています。
164:       |:|:'df'&r1>b-<dq4fq8'e-2g'&|r1(g8a)&r8{gr}'d1f'&r2
165: dfc1(ef)&rgf:||r1e-ga-1g2e-g'f1b-''f2a''fg''fa':|
166: r2.>q5b-q8<(d8e-)&r8q5gq8(g8a-)&r2..a-.g.a-a-1r2.q5fq8
167: >>~20l8b-4<dfr>b-<dfr>b-<df>b-4<df
168: e-4gb-re-gb-re-gb-e-4gb-f4b-<dr>fb-<dl4c>b-agfe-dc>
169: {b-r}a2f11_20<'db-'678@b0,-8192,35'd4fb-'
170:
171: (t4)    /Track 4 -------------------------------------
172: rggggggg_5
173:
174: |:c2cde(eg)rgrg(ga)rgr|f2fga(a<c)rcrc(cd)rcr>:|
175: l16<g2gfedfedcedcc-dc>bab<cdefgabl8
176:
177: _17eee@b0,-180,0erer@b0frg&r2.eee@b0,-180,0erer@b0f
178: re&r2.b-rb+b-rb+rb-r(b-b+)&r4b-rar(ag)&r2..&r*384r1
179:
180: l1rrrrrr
181: |:@z120,,,80,60,20l8o3|:b-<ccc4c4&r1&r1&|r1c>b-<ccc4
182: e-r(ce-)&r>b-(b-<c)&r&r2&r1&r1>:|r2..
183: <c4@v90d2.&(d4>b-)&r1&r1&r2_10|
```

▶やっとわかりました。西川さんが「13日の金曜日」を好きなのは，オープニングにウッキー善司(?)が出てくるからなんだ。
中島　太郎(22)神奈川県

```
184: <<e-dc>b- 14
185:
186: /(t3)とちょっと似ています。
187: <|:c1&r2>a-<e-d1&r2>b-<fe-1&r2|
188: cgf.e-8d.c8>b-8<c8d2.:|¯1018e-fga-r(ab-)rr(ab-)b-4gr1:|
189:
190: @66r2|:r1r1r1r1:|>>¯40(b-<c)&r1rq5>(b-<c)&r
191: q4e-q6(ef)&rq4e-q6(fg)&rq4fe-re-fq8(fg)&r2gfe-c(e-f)&r2
192: rgf4re-4fgrf4g(e-f)g(e-f)e->b-(b-<c)&r2&r
193: q5cq8e-fq8|:g4q5e-q8frcre-r|(ef)&r2e-f:|(dc)&r2
194: >b-<c|:(de-)&rq5cq8(ef)&rq5cq8:|cde-fgg4b-4<{c>b-}q4g
195: q8(b-<c)&r&r1&r1&r2e-4cd4_5q3e-¯5q8c&r1(c-c)&r16>b-.
196: ab-2..{b+b-}a2r(ab-)&rag1&r1cde-dcgr(ef)r(de-)rcrce-
197: fe-fgfe-(ab-)rarfq4g4q8g.b-.<c&r1&r1&r2cde-f4(ef)&r
198:
199: @30_20r14e-f|:'df'&r1>b-<dq4fq8'e-2g'&r1
200: |(g8a)&r8{gr}'d1f'&r2dfc1(ef)&rgf:|
201: e-ga-1g2e-g'f2..b-'{b+b-}8'f2a''fg''fa'
202:
203: ¯10r1r>fb-<(f8g)r8f2r8e-8d8e-8d2c2>b-.<c.(c8d)r8c.>b-.a
204: b-2.g8b-8<e-.>b-.<e-(c-8c)&rd.e-(e-8f)&rg.a(b-8<c)&r>b-.
205: ag&r.(g8a)&r8g8f.e-.dc.>a.<c>b-2.<(b-8<c)r8>'b-1d'_10r8
206: f.e-f
207:
208: /2つ上の段と似ています。
209: |:|:'df'&r1>b-<dq4fq8'e-2g'&|r1(g8a)&r8{gr}'d1f'&r2df
210: c1(ef)&rgf:||r1e-ga-1g2e-g'f2..b-'{b+b-}8'f2a''fg''fa':|
211: r2.>q5b-q8<(d8e-)&r8q5gq8(g8a-)&r2..(a8a-)&rg.q5a-q8
212: a-1&rrrq5fq8b-*768r*576>b-a2f'db-'678@b0,-8192,35'dfb-'
213:
214: (t5)    /Track 5 ---------------------------------------
215: r'dgb''dgb''dgb'132o7@1@p127
216: @D1g@p119f@p111e@p102d@p93c>@p84b@p75a@p66g / <-- ( A )
217: @p57f@p48e@p39d@p30c>@p21b@p12a@p3g@p0f18@D0@3@p44
218:
219: |:4'eg<c''eg<c''eg<c''eg<c':|
220: |:4'fa<c''fa<c''fa<c''fa<c':|
221: |:4'eg<c''eg<c''eg<c''eg<c':|
222: 18|:2'dg<c'g'dg<c'g:||:2'dgb'g'dgb'g:|18
223:
224: |:'g<ce''g<ce''g<ce''g<ce'r'g<ce'r'a<cf'r
225: |'g<cg'&r2.:|'g<ce'&r2.
226: 'dfb-'r'dfb+''dfb-'r'dfb+'r'dfb-'r'd4.fb+''dfb-'r
227: 'cfa'r'c1eg'&r*384 11r
228:
229: _20rr|:5rrr|r:|r2|
230: >18'b-<e-g',0'b-df''a-16<ce-'@1r16
231:
232: ¯20|:4'e-1a-<c',15&r1|'f1b-<d'&r1:|18rr'fb-<d',0rr
233: 'fb-<d''f4b-<d''dgb'r1_10@1
234:
235: / 他のパートは繰り返しています。
236: <r1r2..'e-1gb-',10&r2..@d128¯7<de-de->b-f4e-de-d1&r1
237: r1r1r2rgfe-de-fb-4<cd4&r1&r1&r1r1@3@d0r2
238:
239: o314|:'a<cf',0'gb-<e-''eg<c'r2'e8g<c''e8g<c'rr
240: 'eg<c''gb-<e-''a<cf'r2'a8<cf''a8<cf'|r2:|@1r2_10
241: /(A)の部分と似ています。
242: o7132@D1g@p119f@p111e@p102d@p93c>@p84b@p75a@p66g
243: @p57f@p48e@p39d@p30c>@p21b@p12a@p3g@p0f18@3@D0@p44¯10
244:
245: o414|:8'eg<c'r2'e8g<c''e8g<c'rr'eg<c''gb-<e-''a<cf'
246:    r2'a8<cf''a8<cf'r2|'a<cf''gb-<e-':|
247:   <18'ce-g'_10'ce-g'¯10'cfa'_10'cfa'¯10
248: |:|:4'dfb-''dfb-''dfb-''dfb-':|
249:   |:4'e-gb-''e-gb-''e-gb-''e-gb-':||
250:   |:4'dfb-''dfb-''dfb-''dfb-':|
251:   |: 'cfb-''cfb-''cfb-''cfb-':|
252:      'cfa''cfa''cfa''cfa''ce-g''ce-g''cfa''cfa':|
253:   |:'ce-a-''ce-a-''ce-a-''ce-a-':|
254:  >  'b-<e-g''b-<e-g''b-<e-g''b-<e-g'
255:     'gb-<e-''gb-<e-''b-<e-g''b-<e-g'<
256:   |:'cfb-''cfb-''cfb-''cfb-':|
257:     'cfa''cfa''cfa''cfa'>'b-<e-g''b-<e-g'<'cfa''cfa'
258:
259: _10>|:|: 'dfb-''dfb-''dfb-''dfb-':|
260:        |: 'cfa''cfa''cfa''cfa':|
261:        |: 'dfb-''dfb-''dfb-''dfb-':|
262:        |: 'gb-<e-''gb-<e-''gb-<e-''gb-<e-':|:|
263:        |:4'a<cf''a<cf''a<cf''a<cf':|
264:        |:4'gb-<e-''gb-<e-''gb-<e-''gb-<e-':|
265:        |:4'a<cf''a<cf''a<cf''a<cf':|
266:           'gb-<e-''gb-<e-''gb-<e-''gb-<e-'
267:           'a<cf''a<cf''a<cf''a<cf'
268:        |: 'dfb-''dfb-''dfb-''dfb-':|¯10q7
269: r8         'gb-<e-''gb-<e-''gb-<e-'  ¯10
270:           'a<cf''a<cf''a<cf''a<cf'
271:
272: /2つ上の段と似ています。
273: |:|:|:4'dfb-''dfb-''dfb-''dfb-':|
274:      |:4'e-gb-''e-gb-''e-gb-''e-gb-':||
275:      |:4'dfb-''dfb-''dfb-''dfb-':|
276:      |: 'cfb-''cfb-''cfb-''cfb-':|
277:         'cfa''cfa''cfa''cfa''ce-g''ce-g''cfa''cfa':|
278:     ||: 'ce-a-''ce-a-''ce-a-''ce-a-':|
279:  >      'b-<e-g''b-<e-g''b-<e-g''b-<e-g'
280:         'gb-<e-''gb-<e-''b-<e-g''b-<e-g'<
281:      |: 'cfb-''cfb-''cfb-''cfb-':|
282:         'cfa''cfa''cfa''cfa'
283:  >      'b-<e-g''b-<e-g'<'cfa''cfa':|
284:      |:3'ce-a-''ce-a-''ce-a-''ce-a-'_7:|
285:  >      'b-<e-g''b-<e-g''b-<e-g''b-<e-g'
286:  <   |: 'ce-a-''ce-a-''ce-a-''ce-a-':|
287:     ¯10'cfb-''cfb-''cfb-''cfb-'
288:     ¯11'cfa''cfa''cfa''cfa'
289:      |:4'dfb-''dfb-''dfb-''dfb-':|
290:      |:4'e-gb-''e-gb-''e-gb-''e-gb-':|
291:      |: 'dfb-''dfb-''dfb-''dfb-':|
292:      |:4'cfa''cfa''cfa''cfa':|
293:         'e-gb-'r'c2fa'>'a4<cf''b-<df'678<'b-4df'
294:
295: (t6)    /Track 6 ---------------------------------------
296: r'dgb''dgb''dgb''dgb''dgb''dgb''dgb'_10
297:
298: 11|:'eg<c'&r|'fa<c'&r:|'dg<c''dgb'18
299:
300: |:'g<ce''g<ce''g<ce''g<ce'r'g<ce'r'a<cf'r
301:  |'g<cg'&r2.:|'g<ce'&r2.
302: 'dfb-'r'dfb+''dfb-'r'dfb+'r'dfb-'r'd4.fb+''dfb-'r
303: 'cfa'r'c1eg'&r*384r1
304:
305: 11rrrrrr>|:|:'g<ce-'&r&r&r|'fb-<d'&r&r&r:|
306: 'fb-<d'&r&r&r2|18¯10'b-<e-g''b-df''a-<ce-''fb-<d'
307:
308: _1014|:4'e-1a-<c'&r1|'f1b-<d'&r1:|
309: ¯2018@62r>'fb-'rr'fb-''fb-'r'dg'@49r1_2011:|
310:
311: r2|:6rrrr:|<|:'ceg'&r'cfa'&r'cgb-'&|r'cfa'&r:|
312: r2.¯10'f4b-<d'¯10'f2..a<c'¯10'f8b-<d'¯10'f2<ce-'
313: @e120,90_20@53>q714'g<<ce-''a<<cf'
314: |:'b-1<<df'&r'd<fb-''f<b-<d''b-<<df''b-1<<e-g'&r2|
315: 'b-<<e-a''b-<<e-g''b-1<<df'&r2'f<b-<d''b-<<df'
316: 'b-1<<c''a2<<cf''a<<cg''a<<cf':|
317: 'g<b-<e-''b-<<e-g'<'c1<e-a-'>'b-2<<e-g'
318: 'gb-<<e-''b-<<e-g'<'c1<fb-''c2<fa'>'b-<<e-g'<'c<fa'
319:
320: 11q8_20|:'d<fb-''c<fa''d<fb-''g<b-<e-':|
321: 'a<<cf'&r'g<b-<e-'&r'a<<cf'&r'g2<b-<e-''a2<<cf''d<fb-'
322: ¯10q7r8'g4.<b-<e-'¯14
323:
324: /2つ上の段と似ています。
325: >14'g<<ce-''a<<cf'
326: |:|:'b-1<<df'&r'd<fb-''f<b-<d''b-<<df''b-1<<e-g'&|
327: r2'b-<<e-a''b-<<e-g''b-1<<df'&r2'f<b-<d''b-<<df'
328: 'b-1<<c''a2<<cf''a<<cg''a<<cf':||
329: r2'g<b-<e-''b-<<e-g'<'c1<e-a-'>'b-2<<e-g'
330: 'gb-<<e-''b-<<e-g'<'c1<fb-''c2<fa'>'b-<<e-g'<'c<fa':|
331: r>'e-<gb-'_8'g<b-<e-'_8'b-<<e-g'_8<'c1<e-a-'
332: 'c.<e-a-'>¯b-.<<e-g'<¯c<e-a-'¯6'c1<e-a-'¯4'c2fa'r
333: >¯14'a<<cf'<q811'd<fb-'&r<'fb-'&rrrr_3018q7>
334: 'e-<gb-'r'f2a<<c''c4<fa'q8'd<fb-'681¯14'd<fb-'
335: /                                   ^^^注意!!
336: (t7)    /Track 7 ---------------------------------------
337: @56r1
338:
339: r1r1r1r1r1r1r2.g
340:
341: cr2.r2egcr2.r2cd18d+2ry0,8@126ro4c1&r*384r411rr
342:
343: @98y0,0@u10511rrrrrr|:|:4rr|rr:||c&r
344:
345: |:4rrrr:|:|
346:
347: c2..@57116q4g4g8fe-cr14|:10fe-crr{cc}rrce-f|rr{ff}r2:|
348:
349: q8@64o2@e120,90@u110r_50|:8f8¯8:|_25g8r8a8r8
350: |:r2(a8b-)&rb-8r1r2(d8e-)&re-8|r1r2b-.b-8@9r1
351: q6_15o7c>b-agfe-dc@64o2q8¯25:|
352: @15r1r1r1o4@v120@u120@d3b-2<c2f2@d0@62_15r2
353:
354: o418|:b-rrb-rrb-rarrargarb-rrb-rrb-r<e-rrdrr|cr>:|
355: <e-rfrrfrrfrfrre-rrdre-rre-rre-re-rrdrre-r
356: frrfrrfrgrrarrb-r<crrcrdcr>b-@47@D1116_25>>r2..
357: f¯2c¯2f¯2a¯2<c¯2>a¯1<c¯1f¯1a¯1f¯1a<cfa<cf_29
358:
359: / "b-<df"×5 + "b-<d" + ">b-fd"×5 だよ!!
360: [$]o2|:b-<dfb-<dfb-<dfb-<dfb-<dfb-<d>b-fd>b-fd>b-fd>b-fd
>b-fd|
361: / "e-gb-<"×5 + "e-g" + "e->b-g"×5 だよ!!
362: e-gb-<e-gb-<e-gb-<e-gb-<e-gb-<e-ge->b-ge->b-ge->b-ge->b-
ge->b-g[*]:|
363: <c>fa<cf>a<cfacfab+fa<cf>a<cfacfab+fa<cfa<cf[d.s.]
364: / "a-" + "<ce-a-"×5 だよ!!
365: [coda]o2a-<ce-a-<ce-a-<ce-a-<ce-a-<ce-a-
366: / "g" + "b-<e-g"×5 だよ!!
367: o2gb-<e-gb-<e-gb-<e-gb-<e-gb-<e-g
368: o3c>fa<cf>a<cfacfab+fa<cf>a<cfacfab+fa<cfa<cf@D0
369: q8@64o2@v110@u11014|:r2(a8b-)&rb-8r1r2(d8e-)&re-8|
370: r1r2b-.b-8@9r1q6_15o7c>b-agfe-dc@64o2q8¯25:|@15r1r1r1
371: o5@v120@u12018q8ffff4f4g4f2&r>f411b-b-b+b-
372: @914o7_30r1c>b-agffgab-a2fb-*678b-8.
373:
374: (t8)    /Track 8 ---------------------------------------
375: @15r1
376:
377: r1r1r1r1r1r1r1r1
378:
379: r1r1r1r2g<cegb-1y0,2
380: @128r1r1r2cc¯2cc2y0,3@128rc1
381:
382: 11@56@u120r2.....rrrrr|:|:4rr|rr:||rr
383:
384: |:4r|rrr:|r18@u120<rgdfgdc16>b-16gr1:|116
385:
386: <q4@62r2..g4g8fe-crr6414|:10fe-crr{cc}rrce-frr|{ff}r2:|
387: @e120,30q8@49o4_10rrr'e-g''e-a'11|:'db-'&r'e-b-'&r|
388: 'db-'&r'cb-''ca':|'ca-''e-g''fb-''fa'
389:
390: |:'db-''ca''db-''g<e-':|'a<f'&r'g<e-'&r'a<f'&r
391: 'g2<e-''a2<f''db-'r8'g4.<e-'
392:
393: 'e-4g''e-4a'|:|:'db-'&r'e-b-'&r|'db-'&r'cb-''ca':||
```

▶本番に強くなる方法を誰か教えて。　太田　志輝(17)北海道

```
394: 'ca-''e-g''fb-''fa':|_20'ca-'&r'ca-'
395: _10'c2.a'¯30>'a4<f'<'db-'&r'e-b-'&r'fb-''fa'&r
396: 'e-8b-'r8'f2<c''c4a'
397: _15'b-<fb-'678,216@B0,-8192,35¯15'b-4<fb-',0
398:
399: (t10)   /Track 10 ---------------------------------------
400: cdd{dd}l16<d>bgrbgcr
401:
402: l8|:3crdrcrdcrcdrccd|r:|{bg}
403: 'dc'rdr'dc'rddr'dc''dc'c'db'c'dg'r
404:
405: dlr2dcdcdlr2dcdc l8|:3crdcrcd|crcdrccdr:|
406: r<d>bgcrcdr
407:
408: _10crdcrcdcrcdrcb16g16dr|: crdcrcd crcdrccdr:|
409:                         |:|:8crdcrcd|crcdrccdr:|
410: ¯10|rccdl16dd<dd>bbggccl8
411:
412: _10|:crrcdrrcrcrc d2ccccdrrcrcrcd4d|{bg}:|{dd}
413:    |:crrcdrrcrcrc|d2ccccdrrcrcrcd4d {bg}:|
414: l8'db'c'dg'{cccr}4'dc'rr'dc''dc'r'dc'r2.'c4d':|
415:
416: ¯10r'cb''cb''cg''cg'l4_10|:4'dc''dc''dc'cc'c8d'd8cc:|
417: l16bbbrgggrl8|:16crdrccdc|rcdrccdr:|
418: <'d>b''d>b'>'bg''bg''cd'c'cd'c
419:
420: |:|:3crdrccdcrcdrccdr:|'dc'rdr'dc'rddr|'dc''dc'cdcdr:|
421: 'dc''dc''dc''dc''cb''dc''cg'
422:
423: |:8crdrccdrcrdcrcd|r:|b16g16cd<{d>bgr}4dcd{bg}
424:
425: |:|:|:3crdrccdcrcdrccdr:|'dc'rdr'dc'rdd|r'dc''dc'cdcdr:|
426: <'d>b''d>b'>'bg''bg''cd'c|'cd'c:|{dddd}4
427:
428: |:3crdrccd|crcdrccdr:|{dd}l4ddddddd'dc''d2c'cl16
429: rr_10|:6g_6:||:10¯5g:||:3<ad>bt-5bggt-5:|_30|:d¯3t-7:|
430: |:l8'c+d't-7¯:|r32t60'd4c'b64g64c
431:
432: (t11)   /Track 11 ---------------------------------------
433: r
434:
435: c+rrrrrrr
436:
437: c+rc+r2c+4c+4c+rrrrc+
438:
439: rr2.c+4rrrr|:|:4rrr|r:||r4c+2.
440:
441: |:r2c+ c+|c+c+4a4:|c+4.c+2r8c+4a4
442: |:r2c+|c+ c+c+4a4:|c+2l8rc+rrc+c+rc+l1r:|
443:
444: l4ra|:c+ac+rlrrl|rl:|r2.c+r2l1c+rrr|:7rrr|r:|r2c+4c+4
445:
446: c+|:rrrrrc+r2|a4c+4c+:|c+4c+4
447:
448: ¯20c+_20rrr|:3rrrr:|r8c+4.c+4c+4
449:
450: c+|:4rrrrr|c+r2a4c+4c+:|rr2.a4c+rrrrrrc+4c+2a4c+*678c+
451:
452: (t12)   /Track 12 ---------------------------------------
453: rl
454:
455: |:32f+f+f+f+:|
456:
457: rlrlrlrl|:20f+f+f+f+:|rl
458:
459: <<|:e4e4e4e4:|l8>>>
460:   |: f+f+f+a+ra+rf+ f+f+a+rf+f+a+r:|
461: |:|:8f+f+f+a+ra+rf+|f+f+a+rf+f+a+r:||f+f+a+rr2
462:
463: |:4a+rf+f+r2f+a+rf+r2|f+f+f+f+r2f+a+rf+r2:|rlrl:|
464:
465: r2|:rlrlrlrl:||:16r2f+f+r2f+f+r2:|r2
466:
467: l16<r|:64f+f+f+|f+:|
468:
469: l8>r|:32f+|a+f+f+:|rrrl
470:
471: l16<rr|:128f+f+|f+f+:|
472:
473: l16rr|:28f+f+|f+f+:|f+4f+2f+4
474:
475: /-------------------------------------------------------
476: (p)
```

リスト4　スターフォース アレンジ版用カウンタ表示

```
10:00008E98 00000000  11:00008F16 00000000  12:00008BB0 00000000  1:00008E86 00000000
 2:00008E86 00000000   3:00008E86 00000000   4:00008E86 00000000  5:00008E86 00000000
 6:00008E6E 00000000   7:00008E7A 00000000   8:00008E86 00000000
```

(善)の「勝負はこれからだ」第1回

今月は進藤氏が多忙のため，番外編として私が担当する運びとなりました。ご了承ください。

Z-MUSIC ver.2.0再販のお知らせ

1993年12月に発売された「Z-MUSICシステム ver.2.0」は，おかげさまで完売，増刷の運びとなりました。まだお求めでない方は書店でご注文ください。

Z-MUSICってなんですか

まだ初心者の方たちからときどき問い合わせがあるので，簡単に説明します。

Z-MUSICとは，X68000/X68030上で音楽制御を行うマネジメントプログラムです。X68000の本体同梱のシステムディスクに収録されているOPMDRV3.Xなどと同じ種類のものです。ただ，OPMDRV3.Xに比べて，より一層音源の性能を引き出せるような仕様になっており，簡単に情緒ある演奏データを作成することができます。具体的なスペックとしては，FM音源8チャンネル，MIDI音源16チャンネル，AD PCM疑似8チャンネル(同梱のPCM8.X (C)H.ETOH使用時)が同時に制御可能で，音源のスペック範囲内で同時32パート，256和音の音楽演奏が可能です。

その他，映像同期やゲームなどの効果音制御機能なども装備しており，デモンストレーションや同人ゲーム制作に最適です。また，一切のライセンスを放棄していますから，無断で商用利用が可能なのも特長です。

Oh!X LIVE in '94とは

さて，現在このページのリストは，特に記載がない限りはZ-MUSIC用の演奏データとなっています。さらに，MUSICZ.FNC用という記載がなければ，Z-MUSICで「ZMS」と呼んでいる単なるテキストデータです。

入力は，手持ちのテキストエディタ(シャープ製のED.Xなど)を使用してリストのとおりに打ち込んでください。リスト中の行番号は単なる目安ですので打ち込まないでください。入力が終わったらファイルを保存し，エディタを終了します。

ここで入力に間違いがなければ，

　A>ZP ファイル名

で演奏を開始できます。もし，なんらかの入力ミスがあると，エラーが発生してビープ音が鳴りますので，もういちど確認して正しく入力し直してください。効率のよいエラー撲滅法は「Z-MUSICシステム ver.2.0」に付属のマニュアルをご覧ください。

単純な技術的サポートは，パソコン通信ネットワークNEC・PC-VAN XICLUB(ジャンプコードJXI)で行われていますので，機会があればそちらもご利用ください。

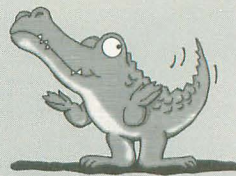
投稿の際には

Z-MUSICを駆使して素晴らしい作品が完成したなら，ぜひともOh!X編集部に投稿してください。お待ちしています。

では，投稿の注意です。

まず，編集部でも最良の演奏状態で曲が聴けるように，投稿のディスクはシステムをインストールして，なるべく自動起動で音楽演奏が始まるようにしてください。MIDI使用時は，音源名や演奏環境などを原稿に添付してください(しかし，最近の投稿ではMIDIを使用した作品が多数を占めているので，掲載には内蔵音源のみの曲が狙い目だったりします)。

また，掲載時は誌面の都合で横64文字の幅に印刷されます。印刷時を想定して，ほかの読者が打ち込みやすいようになるべく見やすいものに仕上げてください。投稿には曲の長さの制限はありませんが，現実問題として，1曲のリストの長さが64字詰めで400行前後までの作品でないと掲載は難しいと思われます。これまでにもリストが長すぎて採用を見送ったなかには素晴らしい作品が多数ありますが，それらについては，機会があったらなんらかの形で発表させていただきたいと思っています。　(西川善司)

(善)のゲームミュージックでバビンチョ

西川善司

ある飲み屋での出来事。

そろそろお開きという，飲み会も大詰め終盤での小事件。

「すいませーん，アイス5つくださーい」

すると，外国人とおぼしき色黒の店員が，

「あの，普通ですか，大盛りですか？」

本来ならこのときおかしいと気づくべきだったのだが，なにぶん酔っ払っていたので，

「おぉ，全部大盛りー！」

と威勢よく頼んだのだが，しばらくして店員が持ってきたのは湯気の揺らめくアツアツの大盛りライスだった。バッフィーーン。

＊　　＊

●サムライスピリッツ/SNK

〜IMAGE ALBUM〜　新世界楽曲雑技団

CD:PCCB-00147　2,500円(税込)

ポニーキャニオン　2/18発売

イメージアルバムというのはたいていはゲームのイメージを壊してくれて，自爆黒焦げ状態のものが多いのだが，コイツは珍しく上質。基本的にオリジナルサウンドのグレードアップ的なアレンジで，ゲームをプレイしたことがある人が聴けば，絶賛間違いなし。トラック1の「タイトルテーマ」からトラック4までを覇王丸，橘右京，柳生十兵衛，服部半蔵と純日本風音楽のノンストップ(!!)。むせび泣く尺八を引き立てる緊張感ある三味線。そして「間」。そう，この「間」さえもひとつの楽器として駆使している感じがする。オリジナルサウンドトラックを買った人は絶対買いだ。

お勧め度　10

●ロックマンX

アルフライラ with 大坪稔明

CD:SRCL2828　3,000円(税込)

ソニー・ミュージックエンタテインメント　3/9発売

カプコンの人気キャラクターのロックマンがついにスーパーファミコンへ進撃。で，このSFC版「ロックマンX」のイメージアルバムがリリースされることに。これもまったく期待していなかったのだが，そう思った自分を懺悔せずにはいられないほどの完成度。全曲，大坪稔明の編曲による完全フュージョンインストアルバム。なんか演奏もアレンジもゲームミュージックの作りでないのでどーもおかしいと思ったら，ゲストミュージシャンがT-SQUAREから本田雅人(SAX)，JIMSAKUから桜井哲夫(BASS)，神保彰(DRUMS)などと豪勢。アルバムのタイトルに「アルフ〜」の文字はあっても，結局この完成度は彼らの手によるものと考えていいだろう。特にお勧めはトラック3とトラック10。

お勧め度　10

●ツインビー

レインボーベルアドベンチャー

CD:KICA-7628〜7629　3,500円(税込)

キングレコード　発売中

なんと，あのコナミの金字塔キャラ・ツインビーがスーパーファミコンでアクションゲームになった。CDは2枚組でディスク1にはゲームのオリジナル全30曲を余すことなく収録。ディスク2にはフルアレンジバージョンが，新キャラ・パステルのおしゃべり日記に挟まれて7曲収録されている。パステルの声は椎名へきる。相変わらず，日常では絶対使わないようなブリッコLANGUAGEで綴られた日記は，ある意味で必聴。

お勧め度　8

●龍虎の拳2/SNK　新世界楽曲雑技団

CD:PCCB-00144　1,500円(税込)

ポニーキャニオン　2/18発売

SNKは商売うまいなぁ。だれもこんな事態は予想しなかった。「サムライスピリッツ」→「餓狼伝説SPECIAL」→「龍虎の拳2」と潜在的ヒット作をこんなにも短期間に重ね当ててくるとは。昔はほこり被ってくすぶっていたNEO・GEOが，いまじゃ至るところのゲームセンターに大量常備されている。やばいぞ本家。夏まで本家の人気は保たないといけないのに……ね。いい加減格闘ゲームにも飽きたといいながら，新しいものが出るたびに結局一目見ずにはいられない。まだまだ続くかこのブーム。13番目の男はアンディだとばかり思ってたのに，実はギースだったとは。この調子だと数年後にはNEO・GEO格闘スペシャルとかいって「餓狼」と「龍虎」のキャラクター総出演のゲームが出てくるんじゃないか？　300メガショーック！

お勧め度　8

文字通り耳よりな話

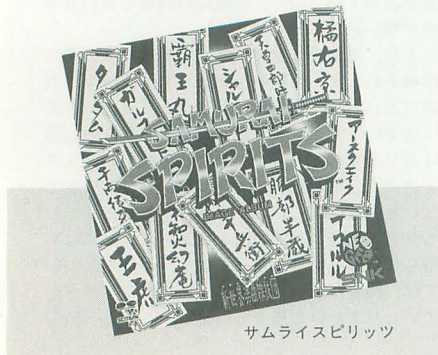
ソニー・ミュージックエンタテインメント「Tel Tel Box」にて，カプコンサウンド・シリーズの試聴サービスが開始された。こりゃCDを買うときの参考になるよ。

試聴方法は，

1)　電話する　東京 03-3624-2662
　　　　　　　大阪 06-700-2662

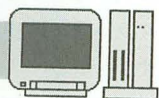
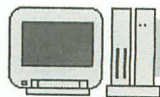
2)　音声ガイダンスに従って「聴かせてコール」へ進む

3)　メニュー番号をダイヤルまたはプッシュする（カプコンサウンドBOXは8009)

[注意]ダイヤル回線の電話の場合は，トーンの切り替えをしてからプッシュすること。

サムライスピリッツ

ツインビーレインボーベルアドベンチャー

龍虎の拳2

X68000←→X68000&その他 ファイル共有の実験と実践 (その6)

仮想ドライバの開発実験PART1.

電机本舗 由井 清人 Yui Kiyoto

今月からいよいよ2台のマシンをつないでのファイル転送の実験に取り掛かります。とりあえず，1台のX68000のSRAMディスクをもう1台から使えるようにしてみましょう。今回はその前編にあたります。

今回よりいよいよ，外部のX68000のディスクを仮想ドライバとして接続する実験を開始します。前回までの解説で，ブロック型デバイスドライバの開発方法の概要が見えてきたと思います。ここでは，このブロック型デバイスを発展させて，外部のパソコン（X68000）と通信接続し，これのドライブを利用する方法を実験していきます。実験は前回のデバイスドライバを利用し，主にC言語により開発していきます。

今回実験するのは，RS-232C通信インタフェイスを利用して接続するものです。そして，接続先のディスクを自分のディスクとして読み書きします。もっとも，今回は相手のSRAMディスクのみに限定して仮想ドライブ化します。

この機能限定は開発が容易であることに由来します。フルスペックに相手側の全ドライブを利用するのであれば，前回作成したデバイスドライバをさらに発展させて，改良する必要があります。前回のSRAMドライバはシングルドライブでした。複数ドライブの制御をするのであれば，各ドライブごとの制御情報を持たしてやらねばなりません。プログラム中の表のうちBPB(BIOS Parameter Block)が，これに相当します。これをドライブごとに持たせて用意する必要があります。

ここでは，実験レポートということで，シンプルなひな型の開発が目的なので，シングルドライブ型を作ります。

仮想ドライバの形態

仮想ドライバの形態はいくつか考えられます。2台のX68000をつなぐまでは同じですが，システムのどのレベ

チェックサムとCRC

通信のチェックで，総じてCRCが優れているといわれていますが，筆者は異なる見解を持っています。

筆者が普段愛用しているのは，排他的論理和によりチェックサムを生成する方法です。排他的論理和を採用しているのは，C言語の記述のしやすさからです。リスト2からの抜粋ですが，

```
    bsc = 0;              ←サム変数初期化
    ptr = (unsigned char *)data;
                          ←転送データをセット
                              ⋮
    for(i=0 ; i<len ; i++)
      {                   ←ループで出力
                              ⋮
        c = *ptr;  ←転送データ1文字取得
        OUT232C(c);←通信へ出力
        bsc^= c;   ←排他的論理和を取る
        ptr ++;    ←次のデータをセット
    }
```

このように，ループ構造の中に簡単に配置でき送信と同時に計算を行えるのがわかります。特に排他的論理和（加算も同じですが）はアセンブラ命令に1行で落ち，計算による遅延を完全に無視できます。おそらく，アセンブラ命令のなかでもかなり高速な命令のはずです。次のように展開されているでしょう。

```
    eor.l d0,$FFF0(A6)
```

特に，条件の厳しい受信側でロスをほとんど無視できる心理的負担の軽さは助かります。

気になる信頼性ですが，1パケットについて文字化けが1回だけ起きるのであれば，100%ハねることができます。ただし，2回起きた場合はサムが1バイトとして，その表現力が256通りですから，偶然正しいサムと同じになる確率は1/256となります。以後，1/128，1/64というふうに低下します。ですから，文字化けが1パケットにつき1回起きるか起きない程度に設定するのがミソでしょう。

ちなみに，1パケット128バイトにサム1バイト，応答コード1バイトととした場合，データ量が130/128に増加しますから，1.56%の転送時間損失です。1パケット64バイトで3.1%，32バイトで6.25%といったところです。

事実上，32バイト以上で利用すれば損失を無視できることがわかります。

また，エラーが起きたときにパケットが小さいほうが再送が楽なので有利かもしれません。

対して，CRCは，$X^{16}+X^{12}+X^{5}+1$の多項式により計算します。このあたりは，筆者の不勉強で詳しく解説できないのが残念ですが，技術評論社刊の「C言語によるアルゴリズム事典」によれば，65535ビット（つまり約8192バイト）に1ビットの化けであれば，一意に補正できるとのことです。

この本記載のサンプルを修正して要所を抜粋したものを次に表します。

チェックサムで，

```
    bsc^= c;     ←排他的論理和を取る
```

の1行だったものが次のようになります。

```
    r^= (unsigned int)(*data)
                        (16 - CHARBIT)
    for(j=0 ; j<CHARBIT ; j++ {
        if(r&0x8000) {
            r = (r<<1)^ CRCPOLY1;
        }
        else {
            r <<1;
        }
    }
```

これは，すべてを計算により行うもので，実際には，計算結果を事前に表にしておき，表参照で高速化したものがあるようです。

筆者の不勉強が原因と思いますが，8Kバイトに1ビットのエラー訂正の自己修正能力というのは，ディスク装置のエラー修正には効果的かもしれませんが，こと，パソコン通信にはあまり向いていないような気がしてしようがありません。

ルで行うかで大きく異なります。図1に各形態を示します。

●並行接続型

双方のX68000に仮想ドライバ用のデバイスドライバを組み込みます。双方とも単体で使用可能で，相手のディスクを自分のディスクとして利用できます。

もっとも高度なスタイルですが，ファイルの共有，ロックなどの問題を考えるといろいろ面倒な問題がありそうです。

たとえば，なにかプログラムを動かしているときに外部のX68000が本機のファイルを読もうとした場合を考えてください。外部のX68000は通信により，本機へディスク読み取りのリクエストを行います。そして，本機でのディスクアクセスはこちらのCPUが行います。ですから，この場合は現在行っているプログラムを一時中断してディスクを読み取る必要があります。

完全なマルチタスク環境であれば，定期的に各プログラムが実行されるので，常時通信を監視するプログラムを実行しておけばよいでしょう。

また，割り込み処理を上手に使用しRS-232Cに入力があれば，現在実行中のプログラムを中断して仮想ディスクの処理を実行するようにしてもよいかもしれません。しかし，マルチタスクであればOSレベルの話ですし，割り込み方式にせよ，高度なプログラミングが要求されます。

●従属接続型

それに対し，こちらは非常に簡単なスタイルです。1台目には仮想ドライブのデバイスドライバを組み込みます。そして，2台目には専用の通信プログラムを実行させておき，常時1台目からの通信リクエストを待つ方式です。この場合，2台目は完全に「高価な専用ディスク」と化します。

動作的には，1台目はデバイスドライバを組み込み起動します。2台目は通常に起動し，そのあとに専用通信プログラムを実行するかたちになります。

今回は，とりあえずこのスタイルを採用します。

従属型でプログラムを作るにあたり，いくつかのタイプがやはり存在します。1台目の，主になるほうは同じなのですが，2台目の従属するほうはいくつかの形式が考えられます。これらの違いは，従属機のディスクをどのように制御するかによります。

ここではSRAMディスクを例にとり説明します。

A) ハード直接操作型

これは，ハードウェアを直接リードライトして，データを制御します。

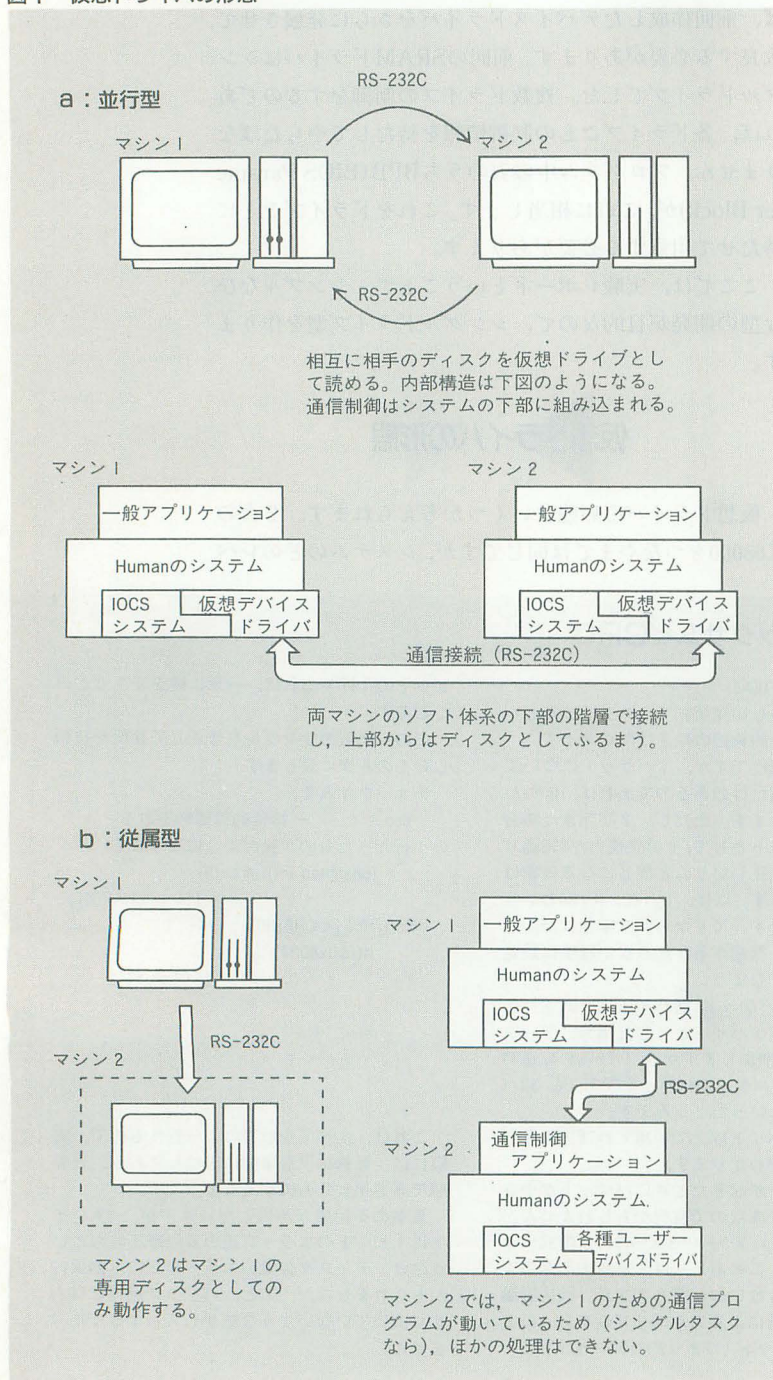
SRAMですと，自分でメモリを読み書きして実行します。

この方式の特徴は，構造が非常に簡単で済むということです。ただし，汎用性はありません。たとえば，ほかのディスクを仮想化しようとした場合，デバイスごとにプログラムを用意する必要があります。OSとのつじつまあわせも自前で行わなければなりません。

B) OS依存型

対して，高水準なのがこちらです。OSの機能を利用し仮想化するディスクへ読み書きを行います。

この方式のよいところは，OSを介してディスクの操作をするので，デバイスごとの相違をOS (Human68k) が吸収してくれるということです。ですから，従属機の各ディスクドライブの読み書きを同じプログラムで行えるというメリットがあります。

図1　仮想ドライバの形態

ただし，デメリットとしては，OSにすでに登録されたデバイスでないと利用できません。ですから，SRAMディスクを主機より仮想ディスクとして利用するのであれば，前もって，SRAMディスクのデバイスドライバを従機の環境設定ファイルconfig.sysへ入れておく必要があります。この方式はすでに利用できるディスクしか仮想ディスクとして，主機へ提供できないのです。

仮想ドライバの全体像

このシステムは基本的には，前回作成したデバイスドライバを改造して作成します。

考え方としては，前回のデバイスドライバを2分割して半々を主機と専従機に分散すると思えばよいでしょう。そして2つに分かれたプログラム同士を通信プログラムにて結合させて動かします。

ですから，主機のほうにはHuman68kとのやりとりを行うドライバのエントリと通信制御プログラムを配置します。

そして，従属機のほうにはSRAMの実体部がありますので実際のSRAMディスク制御プログラムを配置します。もちろん，主機と接続するための通信機構も必要とします。

通信の実際

まず，今回の仕様は仮想SRAMディスクですので，容量が10Kバイト程度ですから，特に高速転送の必要はないでしょう。これより，通信制御はシステムが用意してくれているIOCSコールを利用することにします。

通信仕様は9600bps，データ長8，ストップビット1で行います。

実際には，リスト1のアセンブラプログラムを見てください。これは主機のほうの仮想ドライブのデバイスドライバ本体です。

52行目から始まるサブルーチン“dskstr”はデバイスドライバの事実上の初期化ルーチンです。ここへ，IOCSコールのRS-232C初期化命令を書けばよいことになります。

リスト1の55行目から68行目までを参照してください。ここで，RS-232Cの初期化を行っています。

55行目から62行目まででRS-232Cを9600bps，データ長8ビット，ストップ1，X制御なしに設定しています。

65行目から68行目にかけては，RS-232Cへアルファベットの Xを1文字出力しています。これには深い意味はありません。このようにしておけば，“dskstr”が呼ばれた時点で出力が発生し，デバイスドライバの動作を観測しやすかろうという配慮からです。ちなみに，これにより判明したのは起動後も比較的頻繁にここが呼ばれていたいうことでした。

実際の通信制御に関しては各処理に埋め込めばよいわけです。

通信処理の特殊事情

デバイスドライバの制御はドライバへの制御と状態検出，書き込み，読み込みの3つに大別できます。

通常のドライバですと，ひとつのメモリ空間の上では「陸続き」なわけですから，書き込みですと，ドライバに対して書き込みデータの存在するアドレスを教えてあげるだけでこと足ります。書き込みの場合のリクエストヘッダではCの構造体定義のうち，dmaadrとdmalenの2つのメンバーでデータの所在と長さをドライバに渡しています（リスト2：109行～122行）。

しかし，今回の場合は，実際のドライバは通信ケーブルの反対側にいるので「陸続き」ではありません。ですから，書き込みデータを通信で相手に送ってあげる必要があります。これは，読み込みの場合も同様です。ただ

図2 ハード直接操作型

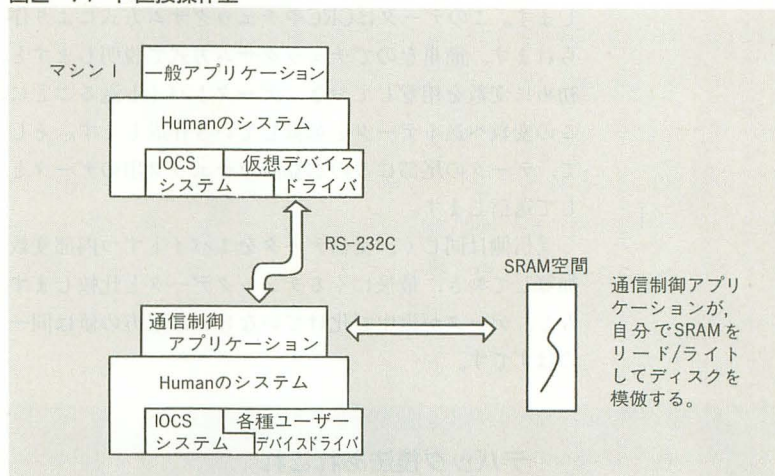
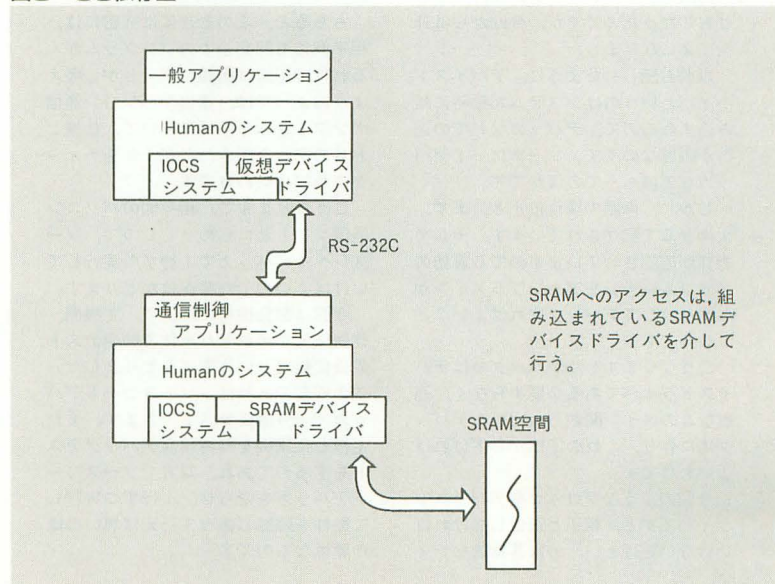



図3 OS依存型

し，このときは相手に送ってもらうわけです。このあたりの制御の考え方が大きく異なるところです。

通信エラーへの配慮

さて，以上のような理由でデータを転送する必要があります。

今回のケースですと，9600bps程度の低速通信，加えて10Kバイト程度のデータですので，エラーに対する配慮は不要に思います。おそらく，きちんとプログラムを組めばノーエラーでしょう。

しかし，ここでは，「やや」本式にプログラムを組んでいきます。

エラーに対する重要項目は次の3点に尽きます。

a) チェックコード

b) リトライ

c) 取りこぼしと同調機構

まず，チェックコードですが，データを送るにあたり尾部に何バイトかのチェック用のデータをつけるようにします。このデータはCRCやチェックサム方式により作られます。簡単なのでチェックサム方式で説明しますと，初めに変数を用意しておき，データ1バイト送るごとに，この変数へ送るデータを加算していき作成します。そして，データの尾部にこの加算値をチェック用のデータとして送信します。

受信側は同じく，受信データを1バイトずつ内部変数加算しておき，最後にくるチェックデータと比較します。もし，データが途中で化けていなければ双方の値は同一のはずです。

もし，一致しなければキャンセルコードを送り，データの転送を再度やりなおします。これがリトライです。

データエラーには，「文字化け」と「取りこぼし」の2つがあります。「文字化け」は文字どおり値が化ける状態ですが，この場合はあまり問題がありません。なぜなら，先のチェックデータで検出できるからです。

問題は「取りこぼし」です。取りこぼしは送信側で，128バイトのデータを送ったとしても，受信側に届いたのが127バイトという具合にデータが落ちている状態です。

多くの場合，データは転送するときに制御情報としてデータ長を最初に送ります。取りこぼしが起きたとき，受信側はこの長さだけ受信を待ちますから，最悪ロックアップが生じます。送信側は正常ないし異常の応答コードを待ち，受信側は残りのデータのくるのを待ち，双方すくみあがる状態になります。

これを回避するために，一定時間データがこないとタイムアウトとみなして転送をやり直すメカニズムを組み込んでやります。

そして，このときに大事なのは，やり直す場合，送信と受信側が等しく噛み合うように，ズレを修正する同調機構を持たせてあげることです。

今回のシステムでは，チェックデータに1バイトのチェックサム（厳密にはサム<加算>ではなく排他的論理和方式）を採用。タイムアウトは，送信側にのみ用意する方式を取りました。

今回採用するパケット方式

リスト4に今回利用するパケット通信の処理プログラムを示します。

3個の基本関数と，2個の内部関数から成ります。

●基本関数

| _rs_buf_clr()

受信用通信バッファをクリア

| blk_in(data, len)

パケット受信を行う，エラー自動回復あり

| blk_out(data, len)

パケット送信を行う，エラー自動回復あり

●内部関数

| blk_in1(data, len)

パケット受信を行う，エラー自動回復なし

| blk_out1(data, len)

パケット送信を行う，エラー自動回復なし

通信は，blk_in(), blk_out()を双方ペアで実行して遂行します。これらの関数は次のかたちでパケット転送を行います。

データ（複数バイト）　チェックコード（1バイト）

RS-232Cへは，上記のかたちでデータが送信されます。このとき，データ長は事前に受信側になんらかのかたち

デバッグ技法あれこれ

前回までは，デバッグに冷汗のかきどおしだったのですが，今回からは非常に楽になりました。

以前お話ししたように，デバイスドライバというのはシステム起動時に組み込まれるので，デバッガなどでの追跡が困難なのです。ソースコードデバッガなどはもってのほかです。

しかし，今回の場合ですと，まず，大半がCで記述されています。そして，動作が確認されていますのでC言語の各サブルーチンをアセンブラメインから切り放してデバッグすればよいことになります。

こうなりますと，デバッグ中はデバイスドライバである必要すらなく，適当なCのメイン関数"main()"をデバッグ用に作り，これの下にぶら下げればよいわけです。

今回のようなブロック型デバイスドライバですと，相手と通信しなければいけない処理というのは基本的なディスクのリード/ライトくらいです。

もちろん，このときには厳密には，相手側にも開発途中のプログラムがくるわけで困難は困難です。しかし考えようによっては，適当なパソコン通信のソフトをぶら下げておいて，仕様どおりのデータがきたかどうかをチェックすればよいわけです。

こうなりますと，相手側のパソコン通信ソフトとにらめっこしつつ，ソースレベルデバッガで1行ずつ実行していけばよいので作業がはかどります。

今回は開発機材の関係で，主機側，従属機側をそれぞれ単体で開発テスト，最後に接続という方式をとりました。この程度であれば，ソースコードデバッガで十分追跡テストできます。また，主機と従属機を同時開発デバッグテストをするのであれ，双方でソースコードデバッガを走らせ，1行ずつ実行して動作を確認しあって行えば思いのほか簡単なものです。

で通知されているものとします。このやり方については後述します。

このパケットは，受信側が1バイトの正常コード（0x00）ないし，エラーを示す異常コード（0x01,厳密には0x00以外）を送信側へ送ることで終了します。

リスト4の136行目から151行目を見てください。送信側は，ここで応答コードがくるのを待っています。もし，5秒以上待ってもこなければタイムアウトを起こしデータ転送をやり直します。

blk_out1()はblk_out()の内部関数で，エラーが発生すると，blk_out()より5回までエラーリトライコールされます。

受信側のblk_in()は基本的にタイムアウトのチェックはしません。取りこぼしによる，永久ループが発生したときには，送信側がタイムアウトを起こして再び送ってくるデータにより，取りこぼしたバイト数の「帳尻」を合わせて受信を異常終了。エラー応答コードを返します。

このとき，送信側はおそらく，2回目以降のデータを連続送信しているはずです。リスト4の122行目を見てください。連続送信しているときに，なんらかの受信が発生したならば，つまり，受信側が応答コードを送ってきたならば，取りこぼしが発生したとみなして，上位のblk_out()へ異常終了し，再度，転送を初めから開始します。

つまり，1回目で取りこぼしが発生したときには，2回目で双方の帳尻を合わせ，3回目で送信と受信が噛み合い正しい転送を行うことになります。

一応，受信と送信ともにリトライを5回まで設定してあります。このようなわけですから，本当は，送信側のリトライ回数を多めに取るのが正解かもしれませんね。

実際のプログラミング

このパケットを利用して，どうやって交信するかを実際のプログラムより解説していきます。

リスト2の255行目から279行目までを参照してください。ディスク交換時に呼ばれる処理mediac()が，ここに記述されています。

まず，268行目でアルファベットの'S'を1バイト送っています。この'S'は転送(Send)の略くらいに思ってください。受信側へこれからデータが送られることを通知します。

次に，269行目で，これから送ろうとするデータの長さをblk_out()で送っています。ここがこのメカニズムのミソです。まず，初めにデータ長を4バイトのデータとしてパケット送信するのです。受信側で同じく4バイトデータの受信をblk_in()で行えばよいでしょう。

そして，次の処理（272行目）でリクエストヘッダそのものをパケット送信しています。受信側には，前の段階でリクエストヘッダの長さが通知されているので問題は起きません。

オーバー9600bpsの高速通信について

今回はSRAM容量が小さいので，あえて高速通信をする必要はありませんでした。

また，仮想ドライバを実現するためには，まだまだ多くの課題があり，RS-232CどころではないのでIOCSコールという標準的な技術を使用した次第です。

しかし，もしも，FD，HD，そしてMOなどといったら9600bps（秒速960バイト）では1Mバイトを転送するのに理論上17分はたっぷりかかることになります。これでは実用に供しません。

X68000の仕様ですと，最高で76800bps。これですと，理論上は2分17秒で転送できるようになります。

もっとも，Human68kが正式にサポートしているのが19200bpsまでですからIOCSなどを使っていたらとてもではないですが，この性能は発揮できません。

こうなりますと，SCCと呼ばれている通信用のLSIを直接叩いてデータ交信を行わないといけません。筆者の経験ですと，38400bpsであれば実際にはノーエラーで転送しているのでまず間違いありません。また，PC/XT互換機で115200bpsが問題なく出ているので76800bpsも問題ないと思います。

ただ，19200bpsを突破しますと，たとえばデータの転送中にマウスを動かしただけで，割り込みが発生してCPUの制御がマウスの矢印移動に占有されて通信データを取りこぼします。また，ディスクアクセスしているあいだも通信が同様の症状を示すなどさまざまな問題が出てきます。これらを回避するためには，通信中はマウスやディスク制御を禁止するなどのキメの細かい処理が必要になってくることはいうまでもありません。

以前，筆者がパソコン通信ソフトの開発を行っているときのことでした。ファイル転送の対向テストを行っている最中，4800bpsの低速通信でデータ取りこぼしが発生したのを覚えています。これは，相方にPC-9801のメジャーなPDSの通信ソフトを利用したのが原因でした。このPDSの移植に携わった人に聞いたところ，ファイル制御はC言語（もしくはPascalだったかもしれない）の512バイトのストリームバッファを利用した標準ファイルライブラリを使っているとのことでした。

C言語でいう，fopen()，fread()，fwrite()は通常の通信制御では利用できません。これらの関数はディスクへの実際のリード/ライトはライブラリが管理しているからです。たとえば，fwrite()ですと，ライブラリの内部バッファが一杯にならないとディスクへの書き込みは行われません。ですから，fwrite()を発行したからといっても実際には書き込まれないのです。

ここで，少し考えてみてください。大量のデータを連続受信している最中にfwriteの内部バッファが一杯になったならば，バッファの一括書き出しが発生し，ほぼ間違いなく通信データの取りこぼしが発生します。これを回避する方法は，データの受信中は実際のディスクへの書き込み命令を行わないことだけです。そして，ディスクへ書き込むときは，相手を待ち状態にして通信を止めておくことです。XMODEMなどのバイナリ転送ですと，パケットを送ると，正しく転送できたかを確認するための応答コードの返送を待ちます。このようなときに受信側はディスクへ書き込んでおき，処理が終わってから応答コードを返えせばよいことになります。さらにいうならば，XMODEMの1パケットは128バイトです。ですから，128バイトごとにディスクへ書き込みを行うのは速度の低下を意味します。現実的には8Kバイト以上の内部バッファを通信プログラムが自前で用意し，適当なタイミングでディスクライトをかけてあげるのが正解です。

まあ，fwrite()を使うのであれば，ライブラリの内部バッファをパソコン標準の512/1024バイトから8Kバイト以上に変更し，かつ，ライブラリが勝手にディスクへ書き出し（これをフラッシュという）しないように，一杯になる前に，安全なタイミングで強制フラッシュをかけておけばよいでしょう。

このような処置をきちんと行えば，C言語と10MHzの68000で76800bpsの高速転送を行えます。

最後に，この処理の結果を従属機からパケット受信してこの処理は終了です。

基本的は，'S'そして，データ長，リクエストヘッダの送信まではほかの処理ともに同じです。ただし，ディスクライト，ディスクリードなどは，実際のデータを向こうに送る。ないし，受信するということから判明するとおり，後処理が大きく異なってきます。

ですが，考え方は同じですので各人で解析してみてください。本質的な違いはありません。

開発環境のクセについて

開発環境はそれぞれ固有のクセがあります。今回の開発でソースコードデバッガscdを使い，今風にソースコードを1行ずつ実行してデバッグを行いました。さて，ここで問題がありました。

scdでデバッグするためにはXCでコンパイルするときに/Nsオプションをつけます。これをつけることにより，実行プログラムにデバッグ情報が付加されます。

この/Nsオプションをつけたところ，それまでコンパイルに成功していたプログラムで，エラーが出るようになりました。つけるオプションにより，コードの生成が変化してくるのです。原因の箇所は次のようなものでした。

```
void sample(void)
{
    ⋮
}
```

注） voidは帰値ないし，引数が存在しないことを表す。

この関数は通常では問題なくコンパイルに成功します。しかし，/Nsのときにはコンパイルエラーになります。これを，

```
void sample()
{
    ⋮
}
```

のように，若干，ANSIのC言語以前の古いスタイルで記述してみたところ問題なくコンパイルされるようになりました。

今回のエラーの厄介なところは，問題点がエラー発生行としてメッセージに出ないことです。筆者は問題点を追跡するために，プログラムの各部を少しずつ削除していき，どこを削除すれば症状が消えるか，患部の“切り分け”を行いました。

読者の方々も，ひょっとすると同様の症状にでくわすことがあるかもしれません。そのようなときは，あわてず患部を“切り分け”るように心がけるとよいでしょう。

その他，補足説明

今回は，主に主機側のプログラムを試作してみました。従属機の制御プログラム試作は次回に譲るものとします。

文中で参照するリスト表を次に示します。

リスト1	d0.s	デバイスドライバ本体
リスト2	d1.c	ドライバの各処理ルーチン
リスト3	d2.s	ドライバの尾部を示すダミー
リスト4	d3.c	通信の入出力ルーチン
リスト5	cx.bat	コンパイルリンク用バッチ

リスト5にコンパイルリンクを行うバッチファイルを示します。このバッチはソースコードデバッグのために，/Nsオプションを使用しています。

今回のドライバは，デバッグを考慮して実行形式のdrvr.xを生成しています。次回，従属機のプログラムの試作が完了した時点でドライバ形式にリンクしなおします。

また，リスト2の139行目から179行目はデバッグ用に利用したメインルーチンです。開発が完了した時点で#define定義されているDEBUGを偽にして，コードより削除する予定です。

リスト1

```
=============== d0.s ==================
  1: ******************************************************************
  2: *  DEVICE DRIVER SAMPLE No1
  3: *  XCプログラマーズマニュアル.P643のSRAMDISKプログラム
  4: *  を改造して作りました。
  5: *
  6: *
  7: ******************************************************************
  8:    .include        doscall.mac
  9:    .include        iocscall.mac
 10:    .text
 11:
 12: .xref       _dskent
 13: .xref       _dskini
 14: .xref       _mediac
 15: .xref       _dskctrl
 16: .xref       _dskout
 17: .xref       _dskotv
 18: .xref       _dskinp
 19: .xref       _notcom
 20:
 21:
 22: .xdef       _d_dat
 23: .xdef       _d_tim
 24: .xdef       _d_dte
 25: .xdef       _inittbl
 26: .xdef       _rdmaxr
 27:
 28:
 29:
 30:
 31: dsktbl:     dc.l    -1
 32:    dc.w    $0000
 33:    dc.l    dskstr
 34:    dc.l    dskent
 35:    dc.b    1,'S_RAM123'
 36: dskreq:     dc.l    0
 37: dskjmp:     dc.l    _dskini
 38:    dc.l    _mediac
 39:    dc.l    _notcom
 40:    dc.l    _notcom
 41:    dc.l    _dskinp
 42:    dc.l    _dskctrl
 43:    dc.l    _notcom
 44:    dc.l    _notcom
 45:    dc.l    _dskout
 46:    dc.l    _dskotv
 47:    dc.l    _notcom
 48:    dc.l    _notcom
 49:    dc.l    _notcom
 50:
 51:
 52: dskstr:     move.l  a5,dskreq
 53:    movem.l d0-d7/a0-a6,-(sp)        * 各レジスタをバックアップ
 54:
 55:                                     * --- rs232c 初期化 ---
 56:    moveq.l #_SET232C,d0             * IOCS コール番号セット
 57:    move.w  #%0100_1100_0000_0111,d1 * stop 1
 58:                                     * parity none
 59:                                     * bit 8
 60:                                     * xon/xoff none
 61:                                     * bps 9600
 62:    trap    #15
 63:
 64:
 65:                                     * --- 'X'をテストで232cへ出力 ---
 66:    moveq.l #_OUT232C,d0             * IOCS コール番号セット
 67:    move.b  #$58,d1                  * 'X'をテスト送出
 68:    trap    #15
 69:
 70:
 71:    movem.l (sp)+,d0-d7/a0-a6        * 各レジスタを復旧
 72:    rts
 73:
 74:
 75: dskent:
 76:    movem.l d0-d7/a0-a6,-(sp)        * 各レジスタをバックアップ
 77:
 78:    move.l  dskreq,a5                * C関数へ引数を渡す
 79:    move.l  a5,-(sp)
 80:    jsr     _dskent                  * C関数をコール
 81:    addq.l  #4,SP                    * 後始末(引数受け渡しで使ったスタックを処分)
 82:
 83:    movem.l (sp)+,d0-d7/a0-a6        * 各レジスタを復旧
 84:    rts                              * Humanへ帰る.さやうならぁ.
 85:
 86:
 87: bpbtbl:
 88:    dc.w    1024
 89:    dc.b    1
 90: bpbftc:     dc.b    1
 91:    dc.w    1
 92: bpbdrc:     dc.w    32
 93: _rdmaxr:    dc.w    $10
```

▶眠気には，「BLACK BLACK」より「STRONG PUNCH」でしょう。夜の通信野郎&プログラマ必須！
鈴木　政宏(20)宮城県

```
 94: bpbfid:    dc.b    $f9
 95: bpbfsz:    dc.b    1
 96:
 97: _inittbl:
 98:     dc.l    bpbtbl
 99:
100:
101:
```

```
102:
103: _d_dat:    dc.b    'SRAM_DISK  ',$08,0,0,0,0
104:     dc.b    0,0,0,0,0,0
105: _d_tim:    dc.b    $A0,$60
106: _d_dte:    dc.b    $E7,$0A
107:     dc.b    0,0,0,0,0,0
108:
109:     end
```

リスト2

```
=============== d1.c ==================
  1: #define    DEBUG   1
  2:
  3:
  4: #include   <doslib.h>
  5: #include   <stdio.h>
  6: #include   <time.h>
  7:
  8: extern     short   d_tim;
  9: extern     short   d_dte;
 10: extern     char    d_dat[];
 11:
 12: /*
 13: extern     struct  BPBPOI  *inittbl;
 14: */
 15:
 16:
 17: extern     char    inittbl;
 18: extern     char    inittblend;
 19: extern     unsigned short  rdmaxr;
 20:
 21:
 22: #define    SRAM    0xed0000
 23: #define    SRAMMD  0xed002d
 24: #define    SYS00d  0xe8e00d
 25:
 26: #define    TIMEOUT 5L
 27:
 28:
 29:
 30: #define    REQLEN  0
 31: #define    UNITCD  1
 32: #define    COMCOD  2
 33: #define    ERRLOW  3
 34: #define    ERRHIGH 4
 35:
 36: #define    MXUNIT  13
 37: #define    DEVEND  14
 38: #define    BPBPOI  18
 39: #define    BDEVNO  22
 40:
 41: #define    DISKID  13
 42: #define    DISKFG  14
 43:
 44: #define    DMAADR  14
 45: #define    DMALEN  18
 46: #define    STAREC  24
 47: #define    GETDAT  13
 48:
 49: char       *_devmes = "C: $ED0400-15K";
 50: char       *_mesini = ")初期化しました ¥n¥r";
 51: char       *_mesnoi = ")初期化しませんでした ¥n¥r";
 52:
 53: int        dskinp();
 54: int        dskout();
 55: int        dskini();
 56: int        mediac();
 57: int        dskpaw0();
 58: int        dskpaw1();
 59: int        dskpaw2();
 60: int        dskotv();
 61: long       _abs();
 62: void _rs_buf_clr();
 63: int notcom();
 64:
 65: /*----- d3.c func -----*/
 66: int        blk_in();
 67: int        blk_inl();
 68: int        blk_out();
 69: int        blk_outl();
 70: void       _rs_buf_clr();
 71:
 72:
 73: struct REQ_HED {
 74:     char    reqlen;
 75:     char    unitcd;
 76:     char    comcod;
 77:     char    errlow;
 78:     char    errhigh;
 79: };
 80:
 81: struct REQ_INI {
 82:     char    reqlen;         /* リクエストヘッダの長さ        */
 83:     char    unitcd;         /* ユニットコード               */
 84:     char    comcod;         /* コマンドコード               */
 85:     char    errlow;         /* エラーコードその1            */
 86:     char    errhigh;        /* エラーコードその2            */
 87:     char    rsv[8];         /* 予約領域                     */
 88:                             /* ---------------------------- */
 89:     char    mxunit;         /* ユニット数                   */
 90:     char    *devend;        /* デバイスドライバ制御プログラムの終了アドレス*/
 91:     char    *bpbpoi;        /*struct BPBPOI *bpbpoi; BPBテーブルアドレス */
 92:     char    bdevno;         /* ブロックデバイス番号(0=A*)    */
 93: };
 94:
 95:
 96: struct REQ_CHG {
 97:     char    reqlen;         /* リクエストヘッダの長さ        */
 98:     char    unitcd;         /* ユニットコード               */
 99:     char    comcod;         /* コマンドコード               */
100:     char    errlow;         /* エラーコードその1            */
101:     char    errhigh;        /* エラーコードその2            */
102:     char    rsv[8];         /* 予約領域                     */
103:                             /* ---------------------------- */
104:     char    diskid;         /* よくわかんない<現在未使用>    */
105:     long    diskfg;         /* よくわかんない 資料ではバイト型*/
106: };
107:
108:
109: struct REQ_RW {
110:     char    reqlen;         /* リクエストヘッダの長さ        */
111:     char    unitcd;         /* ユニットコード               */
112:     char    comcod;         /* コマンドコード               */
113:     char    errlow;         /* エラーコードその1            */
114:     char    errhigh;        /* エラーコードその2            */
115:     char    rsv[8];         /* 予約領域                     */
116:                             /* ---------------------------- */
117:     char    diskid;         /* よくわかんない<現在未使用>    */
118:     char    *dmaadr;        /* 転送バッファ                 */
119:     long    dmalen;         /* 転送セクター数               */
120:     char    starec00[2];    /* アクセスセクタ番号上位2byte   */
121:     long    starec;         /* アクセスセクタ番号           */
122: };
123:
124:
125: struct REQ_CTRL {
126:     char    reqlen;         /* リクエストヘッダの長さ        */
127:     char    unitcd;         /* ユニットコード               */
128:     char    comcod;         /* コマンドコード               */
129:     char    errlow;         /* エラーコードその1            */
130:     char    errhigh;        /* エラーコードその2            */
131:     char    rsv[8];         /* 予約領域                     */
132:                             /* ---------------------------- */
133:     char    getdat;
134: };
135:
136:
137: struct REQ_CHG req_hed;
138:
139: #if DEBUG
140: /***************************************************************
141:     main() テスト用のメインルーチン
142:     エントリー(entry)は全プログラムの入り口と解釈すること
143:     ASMを強引にハンドディスコンパイル
144: ***************************************************************/
145: void main()
146: {
147:    int      sts;
148:    int      s;
149:
150:
151:     for( s=0 ; s<6 ; s++ ) {
152:             req_hed.reqlen = sizeof(struct REQ_CHG);/*リクエストヘッダの長さ*/
153:
154:             switch( s ) {
155:                     case 0:
156:                             sts = dskini( &req_hed );
157:                             break;
158:                     case 1:
159:                             sts = mediac( &req_hed );
160:                             break;
161:                     case 2:
162:                             sts = dskinp( &req_hed );
163:                             break;
164:                     case 3:
165:                             sts = dskctrl( &req_hed );
166:                             break;
167:                     case 4:
168:                             sts = dskout( &req_hed );
169:                             break;
170:                     case 5:
171:                             sts = dskotv( &req_hed );
172:                             break;
173:                     default:
174:                             sts = notcom(&req_hed); /* 未使用命令*/
175:                             break;
176:             }
177:     }
178: }
179: #endif
180:
181:
182:
183: /****************************************************************
184:     diskent 世に言うエントリールーチン (舶来語)
185:     エントリー(entry)は全プログラムの入り口と解釈すること
186:     ASMを強引にハンドディスコンパイル
187: ****************************************************************/
188: void dskent( req_hed )
189: struct REQ_HED *req_hed;
190: {
191:    int      sts;
192:
193:     switch( req_hed->comcod ) {
194:             case 0:
195:                     sts = dskini( req_hed );
196:                     break;
197:             case 1:
198:                     sts = mediac( req_hed );
199:                     break;
```

▶MO買った。(容量が)でかい，FDだと思った(笑)。　　増澤　秀晃(19)長野県

```
200:            case 2:
201:                    sts = notcom(req_hed);              /* 未使用命令*/
202:                    break;
203:            case 3:
204:                    sts = notcom(req_hed);              /* 未使用命令*/
205:                    break;
206:            case 4:
207:                    sts = dskinp( req_hed );
208:                    break;
209:            case 5:
210:                    sts = dskctrl( req_hed );
211:                    break;
212:            case 6:
213:                    sts = notcom(req_hed);              /* 未使用命令*/
214:                    break;
215:            case 7:
216:                    sts = notcom(req_hed);              /* 未使用命令*/
217:                    break;
218:            case 8:
219:                    sts = dskout( req_hed );
220:                    break;
221:            case 9:
222:                    sts = dskotv( req_hed );
223:                    break;
224:            case 10:
225:                    sts = notcom(req_hed);              /* 未使用命令*/
226:                    break;
227:            case 11:
228:                    sts = notcom(req_hed);              /* 未使用命令*/
229:                    break;
230:            case 12:
231:                    sts = notcom(req_hed);              /* 未使用命令*/
232:                    break;
233:            default:
234:                    sts = notcom(req_hed);              /* 未使用命令*/
235:                    break;
236:    }
237: }
238:
239:
240: /*****************************************************************
241:     notcom  未使用命令
242:     ASMを強引にハンドディスコンパイル
243: *****************************************************************/
244: int notcom(req)
245: struct REQ_HED *req;
246: {
247:     req->errlow = 0x03;     /* 下位バイトエラーコード格納    */
248:     req->errhigh = 0x50;    /* 上位バイトエラーコード格納    */
249:     return( 0 );
250: }
251:
252:
253:
254:
255: /*****************************************************************
256:     mediac  メディア交換処理(RAMなので未使用のはず)
257: *****************************************************************/
258: int mediac( req_chg )
259: struct REQ_CHG *req_chg;
260: {
261:    int      sts;
262:    int      n;
263:
264:     req_chg->diskfg = 1L;                          /* エラーコードセット    */
265:     n = (int)(req_chg->reqlen);
266:
267:
268:     OUT232C( 'S' );                                /* 送信開始コード送出    */
269:     if( (sts=blk_out( &n, sizeof(n) ))) {     /* リクエストヘッダ長送信*/
270:             ;
271:     }
272:     else if( (sts=blk_out( req_chg, req_chg->reqlen ))) {/*リクエストヘッダ送信*
/
273:             ;
274:     }
275:     else if( (sts=blk_in( &(req_chg->diskfg), sizeof(req_chg->diskfg) ))) {
276:             req_chg->diskfg = 1L;                  /* エラーコードセット    */
277:     }
278:     return( 0 );
279: }
280:
281:
282:
283:
284: /*****************************************************************
285:     dskinp  ディスク入力処理(?)
286:     ASMを強引にハンドディスコンパイル
287: *****************************************************************/
288: int dskinp( req )
289: struct REQ_RW *req;
290: {
291:    int      sts;
292:    int      n;
293:    int      len;
294:
295:     req->errlow = 0x03;                    /*下位バイトエラーコード格納*/
296:     req->errhigh = 0x50;                   /*上位バイトエラーコード格納*/
297:
298:     n = (int)(req->reqlen);
299:
300:     OUT232C( 'S' );                                /*送信開始コード送出*/
301:     if( (sts=blk_out( &n, sizeof(n) ))) {     /*リクエストヘッダ長送信*/
302:             ;
303:     }
304:     else if( (sts=blk_out( req, req->reqlen ))) { /*リクエストヘッダ送信*/
305:             ;
306:     }
307:     else if( (sts=blk_in( &(req->errlow), 2 ))) { /*エラーコードセット*/
308:             ;
309:     }
310:     else if( req->errlow || req->errhigh ) {
311:             ;
312:     }
313:     else if( (sts=blk_in( &len, 4 ))) {
314:             ;
315:     }
316:     else if( (sts=blk_in( req->dmaadr, len ))) {
317:             ;
318:     }
319:     else {
320:             req->errlow = 0;          /* 下位バイトエラーコード格納*/
321:             req->errhigh = 0;         /* 上位バイトエラーコード格納*/
322:             sts = 0;
323:     }
324:
325:     return( sts );
326: }
327:
328:
329:
330:
331: /*****************************************************************
332:     dskotv  出力&ベリファイ処理(?)
333:     ASMを強引にハンドディスコンパイル
334: *****************************************************************/
335: int dskotv( req )
336: struct REQ_RW *req;
337: {
338:     return( dskout( req ) );
339: }
340:
341:
342:
343: /*****************************************************************
344:     dskout  ディスク出力処理(?)
345:     ASMを強引にハンドディスコンパイル
346: *****************************************************************/
347: int dskout( req )
348: struct REQ_RW *req;
349: {
350:    int      sts;
351:    int      n;
352:    int      len;
353:
354:     req->errlow = 0x03;                            /* 下位バイトエラーコード格納*/
355:     req->errhigh = 0x50;                           /* 上位バイトエラーコード格納*/
356:
357:     n = (int)(req->reqlen);
358:
359:     OUT232C( 'S' );                                /* 送信開始コード送出*/
360:     if( (sts=blk_out( &n, sizeof(n) ))) {     /* リクエストヘッダ長送信*/
361:             ;
362:     }
363:     else if( (sts=blk_out( req, req->reqlen ))) {/*リクエストヘッダ送信*/
364:             ;
365:     }
366:     else {
367:             len = req->dmalen;      /* len = 転送バイト数算出*/
368:             len <<= 10;             /* len = dmalen * 1024 -1*/
369:             len --;
370:
371:
372:             if( (sts=blk_out( &len, 4 ))) { /* データ長送信 */
373:                     ;
374:             }
375:             else if( (sts=blk_out( req->dmaadr, len ))) { /*データ長送信*/
376:                     ;
377:             }
378:             else if( (sts=blk_in( &req->errlow, 2 ))) { /*エラーコードセット*/
379:                     ;
380:             }
381:             else if( req->errlow || req->errhigh ) {
382:                     ;
383:             }
384:             else {
385:                     req->errlow = 0;          /* 下位バイトエラーコード格納*/
386:                     req->errhigh = 0;         /* 上位バイトエラーコード格納*/
387:                     sts = 0;
388:             }
389:     }
390:
391:     return( sts );
392: }
393:
394:
395:
396:
397: /*****************************************************************
398:     dskctrl ディスクコントロール
399: *****************************************************************/
400: int dskctrl( req )
401: struct REQ_CTRL *req;
402: {
403:    int      sts;
404:    int      n;
405:    int      len;
406:
407:     req->errlow = 0x03;                    /* 下位バイトエラーコード格納*/
408:     req->errhigh = 0x50;                   /* 上位バイトエラーコード格納*/
409:
410:     n = (int)(req->reqlen);
411:
412:     OUT232C( 'S' );                                /* 送信開始コード送出*/
413:     if( (sts=blk_out( &n, sizeof(n) ))) {/* リクエストヘッダ長送信*/
414:             ;
415:     }
416:     else if( (sts=blk_out( req, req->reqlen ))) {/*リクエストヘッダ送信*/
417:             ;
418:     }
419:     else if( (sts=blk_in( req, req->reqlen ))) {/*リクエストヘッダ受信*/
420:             ;
421:     }
422:     else {
423:             sts = 0;
424:     }
425:
426:     return( sts );
```

▶どこへいった、タブクリア。　　鈴木　武虎(20)愛知県

```
427: }
428:
429:
430: /*****************************************************************
431:    * dskini  ディスク初期化ルーチン
432: *****************************************************************/
433: int dskini( req )
434: struct REQ_INI *req;
435: {
436:     req->errlow = 0;                    /* 下位バイトエラーコード格納*/
437:     req->errhigh = 0;                   /* 上位バイトエラーコード格納*/
438:
439:     return( 0 );
440: }
```

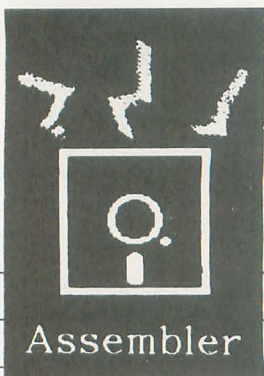
リスト3

```
===============  d2.s  ==================
  1: ****************************************************************
  2: ****************************************************************
  3:     .text
  4:
  5: .xdef        _inittblend
  6:
  7: _inittblend:
  8:    dc.b   0
  9:
 10:    end
```

リスト4

```
===============  d3.c  ==================
  1: #include   <doslib.h>
  2: #include   <stdio.h>
  3: #include   <time.h>
  4:
  5: #define    TIMEOUT 5L
  6:
  7:
  8: int        blk_in();
  9: int        blk_in1();
 10: int        blk_out();
 11: int        blk_out1();
 12: void       _rs_buf_clr();
 13:
 14: /****************************************************************
 15: *  blk_in  複数データを受信
 16: *  受信に失敗したら5回までやりなおす
 17: ****************************************************************/
 18: int        blk_in( data, len )
 19: unsigned char      *data;  /* 転送データ格納アドレス*/
 20: int        len;            /* 転送データ長 */
 21: {
 22:    int       sts;
 23:    int       i;
 24:
 25:    for( i=0 ; i<5 ; i++ ) {
 26:         sts = blk_in( data, len );
 27:         if( sts==0 ) {
 28:                 break;
 29:         }
 30:    }
 31:
 32:    return( sts );
 33: }
 34:
 35:
 36: /****************************************************************
 37: *  blk_in1 複数データ受信
 38: ****************************************************************/
 39: int        blk_in1( data, len )
 40: unsigned char      *data;  /* 転送データ格納アドレス*/
 41: int        len;            /* 転送データ長          */
 42: {
 43:    time_t   tm;
 44:    time_t   tm1;
 45:    time_t   tmx;
 46:    int      bsc;
 47:    int      i;
 48:    unsigned char    *ptr;
 49:    int      c;
 50:    int      sts;
 51:    int      n;
 52:
 53:    sts = -1;
 54:    bsc = 0;
 55:    ptr = (unsigned char *)data;
 56:
 57:    for( i=0 ; i<len ; i++ ) {
 58:            c = INP232C();
 59:            *ptr = c;
 60:            bsc ^= c;
 61:    }
 62:
 63:    c = INP232C();
 64:    _rs_buf_clr();
 65:
 66:    if( c!=bsc ) {
 67:            OUT232C( 1 );
 68:    }
 69:    else {
 70:            OUT232C( 0 );
 71:            sts = 0;
 72:    }
 73:
 74:    return( sts );
 75: }
 76:
 77: /****************************************************************
 78: *  blk_out 複数データを送信
 79: *  送信に失敗したら5回までやりなおす
 80: ****************************************************************/
 81: int        blk_out( data, len )
 82: unsigned char      *data;  /* 転送データ格納アドレス*/
 83: int        len;            /* 転送データ長 */
 84: {
 85:    int       sts;
 86:    int       i;
 87:
 88:    for( i=0 ; i<5 ; i++ ) {
 89:         _rs_buf_clr();             /* 通信バッファクリア*/
 90:         sts = blk_out1( data, len );
 91:         if( sts==0 ) {
 92:                 break;
 93:         }
 94:    }
 95:
 96:    return( sts );
 97: }
 98:
 99:
100: /****************************************************************
101: *  blk_out1        複数データ送信
102: ****************************************************************/
103: int        blk_out1( data, len )
104: unsigned char      *data;  /* 転送データ格納アドレス*/
105: int        len;            /* 転送データ長 */
106: {
107:    time_t   tm;
108:    time_t   tm1;
109:    time_t   tmx;
110:    int      bsc;
111:    int      i;
112:    unsigned char    *ptr;
113:    int      c;
114:    int      sts;
115:    int      n;
116:
117:    sts = -1;
118:    bsc = 0;
119:    ptr = (unsigned char*)data;
120:
121:    for( i=0 ; i<len ; i++ ) {
122:         if( LOF232C() ) {            /* 同調機構、受信側でとりこぼし*/
123:                 return(sts);         /* 強制リターン */
124:         }
125:
126:         c = *ptr;
127:         OUT232C( c );
128:         bsc ^= c;
129:         ptr ++;
130:    }
131:
132:
133:    OUT232C( bsc );
134:    tm = time( NULL );
135:
136:    while(1) {
137:         tm1 = time( NULL );          /* TimeOut チェック */
138:         tmx = tm1 - tm;
139:         if( tmx> TIMEOUT ) {
140:                 break;
141:         }
142:
143:         if( LOF232C() ) {
144:                 c = INP232C();
145:                 if( c==0 ) {
146:                         sts = 0;
147:                 }
148:
149:                 break;
150:         }
151:    }
152:
153:    return( sts );
154: }
155:
156: /****************************************************************
157: *  _rs_buf_clr     通信バッファクリア
158: ****************************************************************/
159: void _rs_buf_clr()
160: {
161:    int      c;
162:
163:    while( LOF232C() ) {
164:         c = INP232C();
165:    }
166: }
```

リスト5

```
===============  cx.bat  ==================
  1: cc /Y /Ns d0.s d1.c d3.c d2.s>tmp.tmp
  2: type tmp.tmp
```

C風のメモリ割り当てルーチン

Murata Toshiyuki　村田 敏幸

プログラムを作る場合，処理によってはさまざまなデータを扱う可能性が出てきます。データの長さや数はいつも一定だとは限りません。そこで，今回はプログラム実行中に必要に応じてメモリを確保したり，解放したりする方法を考えてみます。

今回のテーマは動的なメモリ割り当てだ。が，諸事情により，背景説明はほとんど抜きにして，C言語の標準ライブラリ関数の仕様に準じたメモリ割り当てルーチンを天下りに示し，解説を加えて，さっさと終わることにしたい。

動的なメモリ割り当て

簡単におさらいしておくと，動的なメモリ割り当て(dynamic memory allocation)とは，プログラムの実行過程で必要になったときに初めてメモリを確保し，不要になったら解放するようなメモリの使い方をいう。これに対し，アセンブラの疑似命令.dsでメモリ領域をプログラム中に埋め込む場合のように，メモリをプログラムの実行開始時(実質的にはコンパイル/アセンブル時)に確保して，以後，終了時まで確保しっ放しにしておくのが静的(static)なメモリ割り当てだ。

動的なメモリ割り当ては，事前に用意した大きなメモリブロックを必要に応じて切り出すことで実現される。この動的なメモリ割り当て用のメモリ領域としては，スタックを流用する場合と，ヒープ(heap)と呼ばれる専用の領域を用意する場合とがある。前者はサブルーチン内部で一時的な作業用領域を確保する場面でお馴染みだろう。特別なコードを書かなくても，スタックポインタに対する単純な操作でメモリ領域を捻出できる簡便な方法だ。ただ，スタックはあくまで一時的な作業領域用にしか使えない。サブルーチン中でスタック上に確保したメモリ領域はサブルーチンを抜けるときに解放しなければならず，任意の順序でメモリを確保/解放することはできないのだ。そこで，高水準言語では，テキストに代表される不定長/不定個数のデータを扱う場合など，より複雑な応用に備えて，ヒープからの動的メモリ割り当てが言語仕様に盛り込まれていたり，ライブラリとして提供されていたりする。が，マシン語レベルではそんなものはないので，なければ作ってしまえの方針により，今回のサブルーチン群の出番となる。

天下りの実装例

リスト1に動的なメモリ割り当てを実現するヒープ管理ルーチン群を示す。リスト1にはC言語のライブラリ関数malloc, free, reallocと同名/同機能の3つのサブルーチンが用意されている。mallocは確保したいバイト数をスタックに積んで呼び出すと，ヒープから指定サイズのメモリブロックを切り出して，その先頭アドレスをa0に返す。メモリ不足の場合はccrのNビットを立て，a0に0を入れて戻る。freeはmallocで確保したメモリブロックへのポインタを引数として受け取り，そのメモリブロックを解放して，ヒープに返却する。最後のreallocは確保済

リスト1　HEAP.S

```
 1: *          ふつうのメモリ管理ルーチン
 2:
 3:                 .include        doscall.mac
 4: *
 5:                 .xdef   malloc
 6:                 .xdef   realloc
 7:                 .xdef   free
 8: *
 9: HEAPSIZE        equ     1024*64 *初期ヒープサイズ
10: NALLOC          equ     1024*4  *OSにメモリを要求するときの単位
11:                                 *(1024, 2048, 4096, ...)
12: *
13:                 .offset 0       *メモリブロックヘッダ
14: MSIZE:          .ds.l   1       *ヘッダを含むバイト数
15:                                 *(負なら使用中)
16: MNEXT:          .ds.l   1       *直後のメモリブロック
17: MPREV:          .ds.l   1       *直前のメモリブロック
18:                 .ds.l   1       *未使用
19: SIZEofMHEADER:
20: MCONTENTS:                      *メモリブロック本体先頭
21: *
22:                 .offset 4       *mallocの引数構造
23: SIZE:           .ds.l   1       *確保するバイト数
24: *
25:                 .offset 4       *freeの引数構造
26: MEMPTR:         .ds.l   1       *解放するメモリブロック
27: *
28:                 .offset 4       *reallocの引数構造
29: OLDMEMPTR:      .ds.l   1       *再割り当てするメモリブロック
30: NEWSIZE:        .ds.l   1       *確保するバイト数
31: *
32:                 .text
33:                 .even
34: *
35: allocptr:       .ds.l   1       *次回の検索開始位置
36: heaphead:       .ds.l   1       *ヒープ先頭
37: heaptail:       .ds.l   1       *ヒープ末尾
38: *
39: HEAD            equ     heaphead-allocptr
40: TAIL            equ     heaptail-allocptr
41:
42: *
43: *          メモリブロックを確保する
```

みメモリブロックを再割り当て(サイズを小さくしたり大きくしたり)する。引数はメモリブロックへのポインタと新サイズで，再割り当て後のメモリブロックへのポインタをa0に返す。この戻り値は引数として渡したポインタと一致しているとは限らない。メモリブロックを大きくする場合で，かつ，直後に隣接する空き領域がない場合，reallocは別にメモリブロックを確保し，旧ブロックの内容をコピーして返す。

なお，リスト1ではヒープをHuman68kのDOSコールmallocで確保するので，利用の際には事前にプログラム本体の後ろにある使われていないメモリ領域をHuman68kに返却しておく必要がある。プログラム先頭に，

```
        lea.l     16(a0),a0
        suba.l    a0,a1
        movem.l   a0/a1,-(sp)
        DOS       _SETBLOCK
        addq.l    #8,sp
```

を枕詞のように置いておけばよいだろう。

リスト1ではヒープをリング状の双方向リストの形で管理している。ヒープ中の各メモリブロックはリスト構造でいう“節”となり，前後のメモリブロックへのリンクポインタにより繋がっている。とくに先頭のメモリブロックと末尾のブロックは繋がっているものとして扱う。各メモリブロック先頭には付随情報格納用の16バイトのヘッダ(13～20行)を設けてあり，2つのポインタはここに格納される。残り8バイトのうち，4バイトはメモリブロックの総バイト数(ヘッダの16バイトを含む)を収めるのに使っている。余った4バイトはメモリブロックヘッダのサイズを扱いやすい2^nにするための詰め物だ。

ヒープをリスト構造で管理するうえでは，空きブロックだけをリストに連ねておく流儀と，割り当て済みのブロックも一緒に連ねる流儀とがある。今回採用したのは一緒くた方式だ。この場合，割り当て済みブロックと空きブロックを区別する1ビットのフラグが必要だが，リスト1では独立したフラグは用意せずに，サイズのフィールドを負にすることで割り当て済みブロックを表現するようにしてみた。

では，細部についてはプログラムリストを見てもらいながら解説することにしよう。

●malloc(45～150行)

まず，引数として渡された確保要求バイト数にヘッダサイズを加えて，さらにヘッダサイズの倍数に切り上げている(50～53行)。結果として，メモリの確保はヘッダサイズを基本単位として行われることになる。これは，ヒープ中にヘッダサイズ未満の小さな隙間が生じると処理が繁雑になるので，それを防ぐための処置だ。反面，1バイト確保するのにも32バイト(ヘッダの16バイトを含む)が消費されるので，メモリの使用効率はあまりよくない。

続いて，ワークから検索開始位置を取り出す(55～57行)。メモリを割り当てるたびにヒープを先頭から

```
 44: *
 45: malloc:
 46: SAVREGS            =        d0-d1/a1-a2
 47: SAVSIZ             =        (2+2)*4
 48:                    movem.l SAVREGS,-(sp)
 49:
 50:                    moveq.l #SIZEofMHEADER*2-1,d1   *確保するバイト数に
 51:                    add.l   SIZE+SAVSIZ(sp),d1      *  ヘッダの分を加えて
 52:                    bcs     error                   *  ヘッダサイズの倍数に
 53:                    andi.b  #-SIZEofMHEADER,d1      *  切り上げる
 54:
 55:                    lea.l   allocptr(pc),a2
 56:                    movea.l (a2),a0                 *a0 = d0 = 検索開始位置
 57:                    move.l  a0,d0                   *
 58:                    bne     allocnext
 59:
 60:                                    *ヒープを初期化する
 61: newheap:           move.l  #HEAPSIZE,-(sp) *ヒープ用のメモリを
 62:                    DOS     _MALLOC         *  Human68kに要求する
 63:                    move.l  (sp)+,a1        *
 64:                    move.l  d0,HEAD(a2)     *
 65:                    bmi     error           *
 66:
 67:                    movea.l d0,a0           *ヒープを1個のブロックに初期化
 68:                    move.l  a1,(a0)+        *MSIZE = ヒープ容量
 69:                    move.l  d0,(a0)+        *MNEXT = 自身
 70:                    move.l  d0,(a0)+        *MPREV = 自身
 71:                    movea.l d0,a0           *
 72:                    adda.l  a0,a1           *
 73:                    move.l  a1,TAIL(a2)     *
 74:                    bra     allocnext
 75:
 76:                                    *ヒープを拡大する
 77: expandheap:        move.l  d1,a1           *{
 78:                    addi.l  #NALLOC-1,d1    *確保するバイト数以上で
 79:                    andi.l  #-NALLOC,d1     *  NALLOCの倍数のメモリを
 80:                    movem.l HEAD(a2),d0/a0  *  ヒープに追加する
 81:                    add.l   d1,a0           *
 82:                    sub.l   d0,a0           *
 83:                    movem.l d0/a0,-(sp)     *
 84:                    movea.l d0,a0           *
 85:                    DOS     _SETBLOCK       *
 86:                    addq.l  #8,sp           *
 87:                    tst.l   d0              *
 88:                    bmi     error           *
 89:                    add.l   d1,TAIL(a2)     *
 90:                    move.l  d1,d0           *
 91:                    move.l  a1,d1           *}
 92:
 93:                    movea.l MPREV(a0),a0    *a0 = ヒープ末尾のブロック
 94:                    tst.l   (a0)            *末尾のブロックが未使用なら
 95:                    bmi     addnode         *
 96:                    add.l   d0,(a0)         *  いま追加したサイズを加える
 97:                    bra     allocnext       *
 98:
 99: addnode:           move.l  (a0),d0         *末尾のブロックが使用中なら
100:                    movea.l a0,a1           *  その直後からヒープ末尾までを
101:                    suba.l  d0,a0           *  1個のブロックに初期化する
102:                    move.l  a0,MNEXT(a1)    *
103:                    move.l  a1,MPREV(a0)    *
104:                    movea.l HEAD(a2),a1     *
105:                    move.l  a1,MNEXT(a0)    *
106:                    move.l  a0,MPREV(a1)    *
107:                    movea.l TAIL(a2),a1     *
108:                    suba.l  a0,a1           *
109:                    move.l  a1,(a0)         *
110:                    bra     allocnext
111:
112: allocloop:         movea.l (a0),a0         *a0 = 直後のブロック
113:                    cmpa.l  d0,a0           *リストを一周したら
114:                    beq     expandheap      *  ヒープを拡大して再試行
115: allocnext:         cmp.l   (a0)+,d1        *十分な大きさのブロックが
116:                    bgt     allocloop       *  みつかるまで繰り返す
117:
118:                                    *みつかった
119:                    beq     justfit         *ぴったり
120:
121:                                    *大きめなので分割する
122:                    move.l  -(a0),d0        *d0 = みつけたブロックの大きさ
123:                    sub.l   d1,d0           *確保した分を引く
124:                    lea.l   0(a0,d1.l),a1   *a1 = 分割する位置
125:                    move.l  a1,(a2)         *次回は分割した後半部から検索する
126:
127:                    neg.l   d1              *確保したバイト数を
128:                    move.l  d1,(a0)         *  負にして格納
129:
130:                    movea.l MNEXT(a0),a2    *分割した残りの体裁を整えて
131:                    move.l  a1,MPREV(a2)    *  リストの適切な位置に繋ぐ
132:                    move.l  a1,MNEXT(a0)    *
133:                    move.l  d0,(a1)+        *MSIZE
134:                    move.l  a2,(a1)+        *MNEXT
135:                    move.l  a0,(a1)+        *MPREV
136:
137: mallocdone:        lea.l   MCONTENTS(a0),a0
138:                                            *a0 = 確保したブロックの本体
139:                    moveq.l #0,d0           *正常終了 (N=0)
140:
141: mallocretn:        movem.l (sp)+,SAVREGS
142:                    rts
143: *
144: justfit:           move.l  (a0),(a2)       *次回は直後のブロックから検索する
145:                    neg.l   -(a0)           *確保した印にサイズを負にする
146:                    bra     mallocdone
147: *
148: error:             suba.l  a0,a0           *a0 = 0
149:                    moveq.l #-1,d0          *異常終了 (N=1)
150:                    bra     mallocretn
151:
152: *
153: *          メモリブロックを再割り当てする
154: *
155: realloc:
```

```
156: SAVREGS          =        d0-d1/a1-a2
157: SAVSIZ           =        (2+2)*4
158:                  movem.l SAVREGS,-(sp)
159:
160:                  movem.l OLDMEMPTR+SAVSIZ(sp),a0/a1
161:                  move.l  a0,d0                        *ポインタが0なら
162:                  beq     new                          *  新規確保する
163:
164:                  moveq.l #SIZEofMHEADER*2-1,d1        *確保するバイト数に
165:                  add.l   a1,d1                        *  ヘッダの分を加えて
166:                  bcs     error                        *  ヘッダサイズの倍数に
167:                  andi.b  #-SIZEofMHEADER,d1           *  切り上げる
168:
169:                  lea.l   -SIZEofMHEADER(a0),a0        *a0 = 旧ブロック先頭
170:                  move.l  (a0),d0                      *d0 = －旧サイズ
171:                  bpl     error
172:                  movea.l a0,a1               *a1 = 旧ブロック末尾
173:                  suba.l  d0,a1               *
174:                  movea.l MNEXT(a0),a2        *a2 = つぎのメモリブロック
175:
176:                  add.l   d1,d0               *d0 = 新サイズ－旧サイズ
177:                  bhi     shrink              *新サイズ＜旧サイズなら縮小
178:                  beq     reallocdone         *新サイズ＝旧サイズなら現状維持
179:
180:                                 *拡大する
181: expand:          cmp.l   (a2),d0             *直後に十分な大きさの
182:                  bgt     expand2             *  空きブロックがあるか？
183:                  neg.l   d0                  *
184:                  cmpa.l  a2,a1               *
185:                  beq     cat                 *あればその場で拡大する
186:
187: expand2:         lea.l   MCONTENTS(a0),a1    *別ブロックを確保して
188:                  moveq.l #SIZEofMHEADER,d0   *  旧ブロックを解放する
189:                  sub.l   d0,d1               *
190:                  move.l  d1,-(sp)            *
191:                  bsr     malloc              *
192:                  bmi     reallocretn         *
193:                  move.l  a1,(sp)             *
194:                  bsr     free                *
195:                  addq.l  #4,sp               *
196:
197:                  movea.l a0,a2               *旧ブロックの内容を
198:                  lsr.l   #2,d1               *  新ブロックに転送する
199:                  bra     copynext            *
200: copylooph:       swap.w  d1                  *
201: copyloopl:       move.l  (a1)+,(a2)+         *
202: copynext:        dbra    d1,copyloopl        *
203:                  swap.w  d1                  *
204:                  dbra    d1,copylooph        *
205:                  bra     reallocdone0
206:
207:                                 *縮小する
208: shrink:          neg.l   d0
209:                  tst.l   (a2)                *直後が空きブロックでなければ
210:                  bmi     split               *  このブロックを単に分割する
211:                  cmpa.l  a1,a2               *
212:                  bne     split               *
213:
214: cat:             add.l   (a2),d0             *直後の空きブロックを仮に併合する
215:                  movea.l MNEXT(a2),a2        *←ccr不変
216:                  bne     split               *
217:                                 *併合したらぴったり（拡大時のみ）
218:                  neg.l   d1                  *直後の空きブロックを併合する
219:                  move.l  d1,(a0)             *
220:                  move.l  a2,MNEXT(a0)        *
221:                  move.l  a0,MPREV(a2)        *
222:                  move.l  a2,allocptr         *
223:                  bra     reallocdone
224:
225: split:           lea.l   0(a0,d1.l),a1       *分割する
226:                  neg.l   d1                  *
227:                  move.l  d1,(a0)             *
228:                  move.l  a1,MNEXT(a0)        *
229:                  move.l  a1,MPREV(a2)        *
230:                  move.l  a1,allocptr         *
231:                  move.l  d0,(a1)+            *MSIZE
232:                  move.l  a2,(a1)+            *MNEXT
233:                  move.l  a0,(a1)+            *MPREV
234:
235: reallocdone:     lea.l   MCONTENTS(a0),a0
236:                                              *a0 = 確保したブロックの本体
237: reallocdone0:    moveq.l #0,d0               *正常終了（N=0）
238:
239: reallocretn:     movem.l (sp)+,SAVREGS
240:                  rts
241: *
242: new:             move.l  a1,-(sp)
243:                  bsr     malloc
244:                  addq.l  #4,sp
245:                  bra     reallocretn
246:
247: *
248: *        メモリを解放する
249: *
250: free:
251: SAVREGS          =        d0-d1/a0-a2
252: SAVSIZ           =        (2+3)*4
253:                  movem.l SAVREGS,-(sp)
254:
255:                  move.l  MEMPTR+SAVSIZ(sp),d0
256:                  beq     freeretn            *MEMPTR = 0 なら即リターン
257:                  movea.l d0,a0
258:                  lea.l   -SIZEofMHEADER(a0),a0  *a0 = 解放するブロック
259:                  move.l  (a0),d0
260:                  neg.l   d0                  *d0 = 解放するバイト数
261:
262:                  lea.l   allocptr(pc),a2
263: chknext:         movea.l MNEXT(a0),a1        *直後に隣接するブロックが
264:                  move.l  (a1),d1             *  空きブロックなら併合する
265:                  bmi     chkprev             *
266:                  adda.l  d0,a0               *
267:                  cmpa.l  a0,a1               *
```

調べるのは効率が悪いので，割り当てたメモリブロックの位置(正確にはその直後)を35行で用意したワークallocptrに覚えておき，次回の空き領域検索時にはそこから調べ始めるようになっている。それが0だった場合は1回目の呼び出しなので，ここで初めてヒープを初期化する(61～73行)。初期化といっても，Human68kに要求して適当なサイズ(9行の定数HEAPSIZE)のメモリを確保し，ヒープ全体を1個のメモリブロックだけからなるリング状の双方向リストの形に整えるだけだ。節が1個しかないので，メモリブロックのサイズはヒープ全体のサイズと等しく，直前のメモリブロックも直後のメモリブロックも自分自身となる。

ヒープを初期化したら(2回目以降の呼び出し時には直接)，メインループ(112～116行)に飛び込む。このループでは，リストを順に辿って，確保要求サイズ以上の大きさの空きメモリブロックを探している。ブロックが割り当て済みか空きかの判定はブロックサイズの比較に埋もれている点に注意しよう。確保済みメモリブロックはブロックサイズを負にしてあるので，符号付きで比較すれば“とても小さなメモリブロック”として自動的に弾かれる。

十分な大きさのメモリブロックが見つかったら，ループを抜けて119行にくる。見つけたメモリブロックの大きさが確保要求バイト数と等しい場合はさらに144行に飛んで，確保した印にブロックサイズを符号反転し，メモリブロックの正味内容(ヘッダの直後)をa0に入れて返す。そうでなければ，見つけたメモリブロックは大きすぎるので2つに分割し，前半部を確保して返し，後半部は空きブロックとしての体裁を整えてリストに繋ぐ(122～135行)。次回の検索はこの分割した後半部から始めることになる。

リストをリング状にしてあるので，もし，十分な大きさの空きメモリブロックがなかった場合には，検索がリストを1周し，検索開始位置に戻ってくる。この場合は，すかさずHuman68kにさらにメモリを要求してヒープを拡大し(77～110行)，再試行する。ヒープの拡大は10行の定数NALLOC単位で行う。ただし，確保要求バイト数が極端に大きい場合にヒープの拡大～再試行のサイクルを何度も繰り返さなくても済むよう，Human68kに対しては確保要求バイト数を超えるNALLOCの最小の倍数をまとめて要求している。ここで，リスト1ではNALLOCは4096と定義されているので，79行はワード演算で済むのだが，NALLOCをもっと大きな値に定義し直して使う場合を想定して，ロングワード演算とした。ヒープの拡大はそうたびたび行われるものではないので，このことによる実行時間の増加は問題にはならない。

新たにヒープに追加したメモリをリストに繋ぐ際には，ヒープ末尾のブロックが割り当て済みか，空きかによって処理を振り分ける。割り当て済みだった場合は，追加分を単独のメモリブロックとしての体裁に整え，リスト末尾に繋ぐ(94～96行)。空きだった場合は，ヒープが無意味に分断されないよう，

追加分をその空きブロックに併合する(99~109行)。

●free(250~302行)

本来，フラグによって空きブロックと割り当て済みブロックを区別するという方式では，フラグを倒すだけでメモリブロックを解放できる。しかし，それだけではヒープが徐々に細かな空きブロックに分割されることになり，やがては，連続する空きメモリがあるのに，それぞれが小さなメモリブロックになっているので，一度に確保できる容量が制限されるという事態に陥る。そこで，適当なタイミングで，隣接する空きメモリブロックをひとつのブロックに併合しなければならない。併合操作はいつ行ってもよいのだが，リスト1ではfreeでメモリブロックを解放したらすぐ行うようになっている。

freeでは，まず，引数として渡されたポインタが0の場合を弾く(255~256行)。今回のプログラムではエラーチェックをほとんどしていないのだが，Cの仕様がこうなっているのでそれに従った。実際，応用上も，freeに0を渡すことができると楽ができる場合が多い。

以下，262~298行では隣接する空きブロックがあるかどうか調べ，あれば併合している。やや冗長だが，263~278行で直後の空きブロックを併合し，280~296行で直前の空きブロックを併合するという2段構成だ。この部分では，必要に応じてワークallocptrを更新している点に注意しよう。これを怠ると，allocptrがメモリブロックの途中を指したままになる可能性がある。

●realloc(155~245行)

Cの仕様では，引数のポインタが0の場合は新たにメモリブロックを確保する(malloc相当)ことになっているので，真っ先にチェックする(161~162行)。本当は，Cではもうひとつ例外規定があり，新サイズが0なら引数で渡されたメモリブロックを解放する(free相当)のが正しいのだが，そんな呼び出し方は誰もしないという決めつけにより，こちらの仕様は採用していない。リスト1のreallocは新サイズが0の場合は，正直にメモリブロックの正味容量を0にして，ヘッダだけのブロックを返す。

164~178行では例によって，確保要求サイズをヘッダサイズの倍数に切り上げてから，若干の下準備ののち，メモリブロックを小さくするのか大きくするのかで処理を振り分ける。小さくする場合は208行以下で，必要に応じてメモリブロックの併合/分割を行う。大きくするときには，直後に十分な空き領域があればその場で拡大する。この操作は，直後の空きメモリブロックをいったん併合してから再分割することで実現されるが，これはメモリブロックを小さくする場合と変わらないので，同じルーチンが兼用されている。その場で拡大できなければ，mallocを呼び出して新たにメモリブロックを確保し，そこへ旧ブロックの内容を転送する(187~204行)。メモリブロックサイズは16の倍数だということがわかっているので，転送はロングワード単位で行う。

```
268:             suba.l   d0,a0            *←ccr不変
269:             bne      chkprev          *
270:                                       *
271:             cmpa.l   (a2),a1          *←必要ならallocptrを更新
272:             bne      catnext          *
273:             move.l   a0,(a2)          *
274:                                       *
275: catnext:    add.l    d1,d0            *
276:             movea.l MNEXT(a1),a1      *
277:             move.l   a1,MNEXT(a0)     *
278:             move.l   a0,MPREV(a1)     *
279:
280: chkprev:    movea.l MPREV(a0),a1      *直前に隣接するブロックが
281:             move.l   (a1),d1          *  空きブロックなら併合する
282:             bmi      freedone         *
283:             adda.l   d1,a1            *
284:             cmpa.l   a1,a0            *
285:             bne      freedone         *
286:             suba.l   d1,a1            *
287:                                       *
288:             cmpa.l   (a2),a0          *←必要ならallocptrを更新
289:             bne      catprev          *
290:             move.l   a1,(a2)          *
291:                                       *
292: catprev:    add.l    d1,d0            *
293:             movea.l MNEXT(a0),a0      *
294:             move.l   a0,MNEXT(a1)     *
295:             move.l   a1,MPREV(a0)     *
296:             movea.l a1,a0             *
297:
298: freedone:   move.l   d0,(a0)          *いまできた空きブロックの
299:                                       *  サイズを格納
300:                                       *N=0
301: freeretn:   movem.l (sp)+,SAVREGS
302:             rts
303:
304:             .end
```

性能評価

リスト1はあまり速度性能にはこだわっていないプログラムなのだが，それなりに実用にはなるはずだ。比較できるものがなかったので，Cから呼び出せるように微修正したうえで，リスト2[1)]の処理時間をXCのライブラリ関数と比べてみた範囲では十分速かった。もっとも，XCのmalloc/freeはかなり遅いので，速くて当然という話もある。実際に比べてみたわけではないが，あれより遅いのは『グラフィックス編』の単行本におまけに付けたヤツぐらいのものだろう(あれはまったく練られていなかった)。

次回の予定は未定。来月かどうかも未定。とりあえず，そろそろ“3冊目”のシーズンなのでそっちをまとめるのが先だ(その前にトンネルを抜けなきゃね)。

1) リスト2では，1~100バイト長のM個(仮に512個)のメモリブロックをmallocで確保しておき，そのうち1つをランダムに解放して，代わりに別のメモリブロックを確保するという操作をN回(仮に10000回)行っている。

リスト2 HEAPTEST.C

```
 1: #include    <stdlib.h>
 2: #include    <time.h>
 3:
 4: #define N   512
 5: #define M   10000
 6: #define ALLOCTEST()  malloc(rand() % 100 + 1)
 7:
 8: int main()
 9: {
10:     int i, n;
11:     void *array[N];
12:
13:     srand(time(NULL) & 0xffff);
14:
15:     for (i = 0; i < N; i++)
16:         if ((array[i] = ALLOCTEST()) == NULL)
17:             abort();
18:     for (i = 0; i < M; i++) {
19:         n = rand() % N;
20:         free(array[n]);
21:         if ((array[n] = ALLOCTEST()) == NULL)
22:             abort();
23:     }
24:     for (i = 0; i < N; i++)
25:         free(array[i]);
26:
27:     return 0;
28: }
```

“実戦！”ゲーム作りの

KNOW HOW

〈基本編　その2〉

BGのマッピング処理(簡略版)

Taguchi Atsushi 田口　敦

実戦的なゲーム作成のためのアルゴリズム講座第2回です。今回は画面サイズより大きなパターンデータを扱うための，スプライトBGを使ったマップ処理の基本について解説します。

◆はじめに

さて，この号が出る頃には冬の寒さもやっとピークを過ぎたって感じになっているのでしょうか。コンシューマの世界では寒気などにはおかまいなく，次期ゲームマシンの開発競争は，ますます加熱してきたようですね。どれもなかなかの性能でソフト開発側としてはどのマシンで開発するか頭を痛めそうですが，私としては好物の食べ物（柿の種，スイカ，芋など）を同時に出された気分でいまからとってもワクワクしています。ただ，68000などのCISCプロセッサプログラマにとっては少々悲しいことですが，次期ゲームマシンに使われるCPUはすべてRISCプロセッサなんですね。

これも時代の流れといっちゃあそれでおしまいなんですが，ゲーム作りの基本はRISCだろうが，CISCだろうが変わりません。RISCで生き残るためにCISCでしっかり勉強しておきましょう。

ちなみに，RISCではアセンブラができないのではないかと悲観されている方に朗報。確かにRISCでは低級言語での開発がまともにできません。なぜかというとRISCの場合，パイプライン管理やキャッシュなどの制御も自分で管理しなければならないからです。しかも命令数もCISCに比べ極端に少ないため最適化が難しく，ソースの視認性が非常に悪いのです。

ではアセンブラプログラマに対してどのように対応しているのかというと，アセンブラ自体がすでに高級言語と化しているのです。これはどういうことかというと，アセンブラ（この場合のアセンブラとはソフトのこと）がプログラマの組んだソースをプロセッサがうまく駆動するように最適化してしまうのです。つまりニーモニックとバイナリコードが必ずしも一致しないのです。しかも少ない命令を補うため，RISCアセンブラにはCISC同様にプログラムが組めるよう，マクロ命令というものが用意されています。これはRISCの命令をいくつか組み合わせ，複雑な処理を行えるようにしたものです。

まあそんなわけで68000のアセンブラを勉強しても損はないのでいけるところまでいっちゃいましょう。

◆今回の内容

おっと，そろそろ本題に入らないと。

前回1月号では，ラスタースクロールの利用法として代表的なものを紹介しました。

初回としては少々難しかったかもしれませんが，とりあえずあの程度のプログラムが自作できるようになれば，割り込みの基本をマスターしたことになるんじゃないかなと思いますが，どうでしょうか。

今回はいくらか慣れてきたということでもっと需要度の高いBGのマッピング処理について勉強したいと思います。

「BGのマッピング処理ってなに？」という方も結構多いと思いますが，早い話が「BGの実画面の大きさを疑似的に拡張する」ことです。

シューティングゲームやアクションゲームでは，横・縦・全方向スクロールを問わず非常に広い画面が使われているのがわかると思います。どう考えたって表示画面の2×2倍の大きさの実画面に収まりそうもありません。そこでマッピング処理を行い，疑似的に実画面の大きさ以上の画面を使用可能にするわけです。

◆マッピング処理の実際

実画面以上の画面を扱うにはどうしたらよいか。これはちょっと考えればすぐに思いつきます。つまり，メモリ上に実画面以上の大きさのデータをマップ情報として持っておき（以下，メモリ上に置く元データのことを単にマップといいます），必要に応じてBGなどの画面にコピーしてやればよいわけです。

まあここまでは誰でも考えるのですが，いざ，アルゴリズムを考えてみると画面をコピーするタイミングや場所など，悩むことは結構あります。

あ，そうそう。もうひとつ非常に重要なことがありました。

X680x0には標準的なマップエディタがありません。いくつかの市販やフリーウェアのスプライトエディタの中にマップエディタが組み込まれていますが，どれもデータのフォーマットが違うので簡単に利用できないということですね。

そこら辺の管理はとりあえず読者の皆さんに任せましょう。フォーマットが違うといってもそう大差はないと思います。

将来的にOh!X上でゲーム作成の標準的な環境を作成できたらなぁ，などと思っています。一応少しずつ進めているので，あまり期待しないよう，待っていてください。

◆アルゴリズム

今回は簡略版ということで，比較的簡単な横スクロールを題材にしたいと思います。

現段階でサンプルに使用できるマップエディタがないので，とりあえずグラフィック画面に適当なものを描いて，それをマップデータとしたいと思います。

使用する割り込みも今回はひとつだけ。前回，割り込み動作が読み切れずにくじけた方は，今回のサンプルプログラムの割り込み動作を解析してみてください。割り込み内の動作は至極単純に作ってあるので，かなりわかりやすくなっていると思います。

しつこいようですが，割り込み処理はリアルタイムゲームの基本なのでしっかりマスターするようにしてください。

さて，実際のアルゴリズムですが，まず図1を見てください。

実画面を疑似的に拡張するわけですから，ユーザーのわからないところで実画面を書き換えなければいけないのがわかると思います。

昔のファミコンなどで，縦スクロールする際に画面の最上段でチラチラ書き換えするのに見覚えがある人も多いと思います。

あれはファミコンの画面のモードによっては，縦方向の実画面の大きさが表示画面と同じ大きさしか持っていないので（違っていたらごめんなさい），ユーザーに見えないところで書き換えることができなかったのです。

X680x0の場合，どの画面モード（裏モードを除く）でも必ず実画面のほうが表示画面より大きくなっているので，このような不都合は発生しません。

ふつう横スクロールの場合，これからスクロールして見える直前の場所を書き換えるのが定石です。

X680x0は256×256ドットモードの場合，BGのキャラクタの大きさが8×8ドットなので，8ドットスクロールするごとに書き換えれば問題はないでしょう。

ちなみに512×512ドットモードの場合，キャラクタの大きさは16×16ドットなので，当然16ドットスクロールするごとにこれから表示されるエリアにデータを書き込むわけです。

さて，実際のデータの読み込みや書き込みの算出法です。

横スクロールの場合，重要なのはX方向の座標から読み込み＆書き込みのアドレスを算出することです（表1）。

BGのキャラクタの大きさがモードによって違うので気をつけなければいけません。私はそれをすっかりド忘れしていてサンプルを作るときにはまりました（ああ，自己嫌悪）。

式の説明はいらないと思います。サンプルプログラムと併用して確認してみてください。

X座標のスタートアドレス算出法がわかったら次にY軸の読み書きアドレス座標算出です。これは簡単で，読み込む側はマップのX方向の最大キャラクタ数に2を掛けたもの，書き込むBGは64キャラと固定なので，それに2を掛けたものを先ほどの式で計算したアドレスにY軸の座標分（今回は全部で32個）だけ足してあげればOKです。

ところで，これだけではまだ完全ではありません。もうひとつの要素が必要です。それは「右にスクロールするときは右側を書き換える。左にスクロールするときは左側を書き換える」というものです（図2）。

これは単純に考えて右側を読み書きするときは先ほど計算したX座標のスタートアドレスに33×2を足して，左側を読み書きするときは－1×2を足せばよいわけです。

図1　書き換えのタイミング

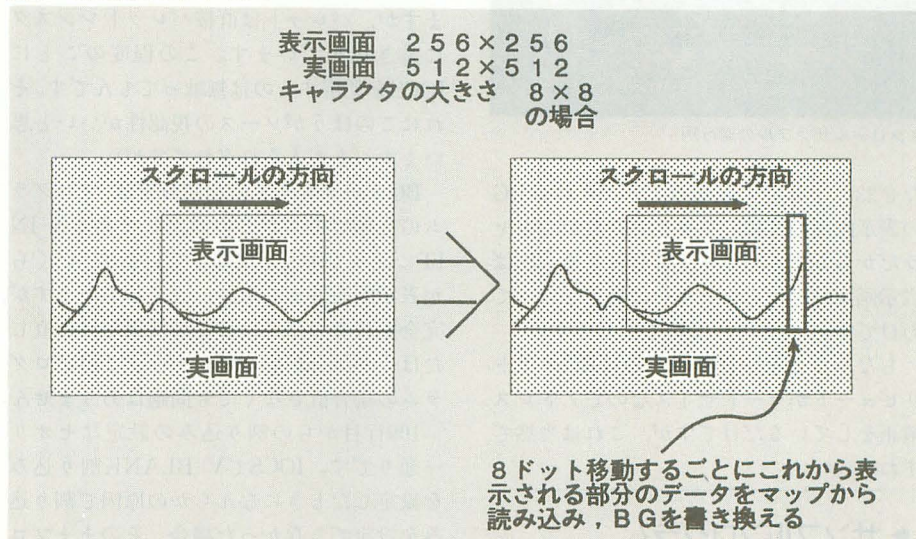



図2　左右移動時の書き換え場所

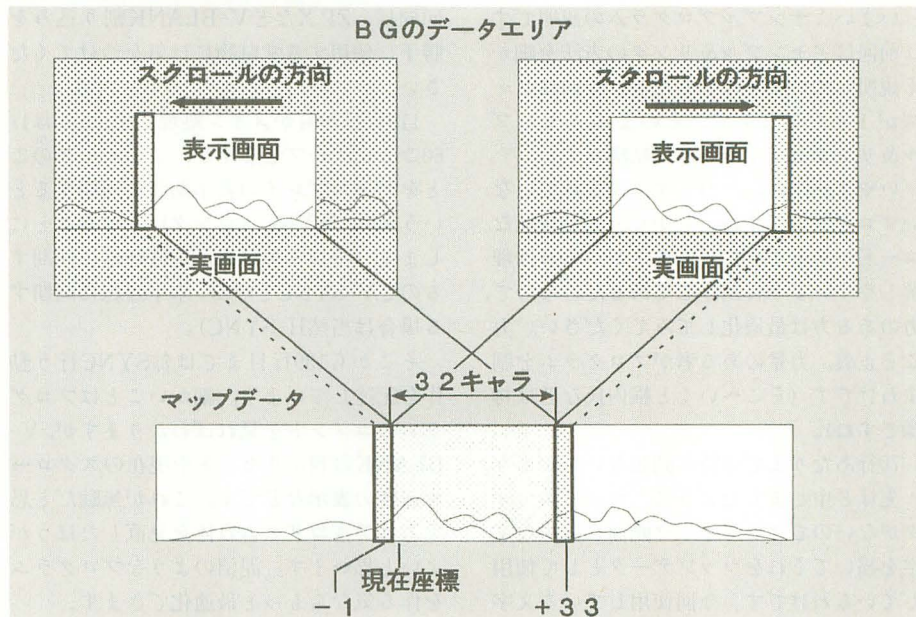



表1　アドレスの計算式

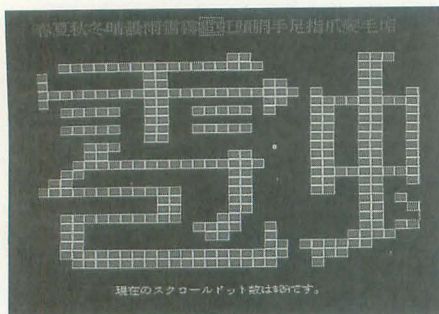
●実画面の大きさ512×512ドットの場合

読み込むスタートアドレス＝マップデータのスタートアドレス
　＋（現在のX座標÷8×2）%（マップのX方向の最大キャラクタ数×2）

書き込むスタートアドレス＝BGエリアのスタートアドレス　$EBC3000（BG0）or $EBE000（BG1）
　＋（現在のX座標÷8×2）%64キャラクタ×2

●実画面の大きさ1024×1024ドットの場合

読み込むスタートアドレス＝マップデータのスタートアドレス
　＋（現在のX座標÷16×2）%（マップのX方向の最大キャラクタ数×2）

書き込むスタートアドレス＝BGエリアのスタートアドレス　$EBC3000（BG0）or $EBE000（BG1）
　＋（現在のX座標÷16×2）%64キャラクタ×2

スクロールサンプルの実行例

なぜ33×2を足すのかって？　それはBGの表示画面のX軸のキャラクタ数が32キャラだからです。33キャラ目を書き換えれば表示画面外だからユーザーに見えないってわけです。−1×2ってのも同じです。

ちなみに2を掛けているのはBGのアトリビュートがワードサイズなのでアドレス補正をしているだけですが，これは当然ですね。

◆ サンプルプログラム

いよいよサンプルプログラムの説明です。

前回はアセンブル&リンクの方法を細かく説明していませんでしたが，今回はソースが1本だけなので，そのままアセンブル&リンクしてくだされば結構です。

いやぁー，あいかわらず洗練されていないプログラムですね。しかし，一見無駄なコードがあるように見えますが，それは理解しやすいように制作してあるだけなので，力のある方は最適化してみてください。力こそ正義。力量のある者がプログラムを制すわけです（そこへいくと横内氏などは神様ですね）。

70行あたりまでは特に問題ないでしょう。

先ほど申しましたように，マップエディタがないのでグラフィック画面に適当な文字を描いてそれをマップデータとして使用しているわけです。今回使用している文字は特に意味がありませんが，Oh!Xにしては当たり障りのない文章なので気にいらない方は書き直してください。279行目にある文字列を変更してもらえばOKです。

“MAP_ADR”という変数がマップの先頭アドレスを指しています。マップエディタをお持ちでマップを制作した方は，ここでセットしている$C00000をデータロードしたアドレスに替えてもらえればそのまま使用できます。その際に気をつけなければならないのが，たいていのマップデータは先頭にいくつかのヘッダデータがついているのでそれを読み飛ばすようにしてください。

あとは制作したマップのX軸のキャラクタ数を変数“MAP_XMAX”にセットしてもらえれば，これもOKです。

71行目からはPCGの設定&BGデータエリアの初期設定です。

PCGの設定は無難にIOCSを使用していますが，パレットは直接パレットレジスタに書き込んでいます。この程度のことにIOCSを使用するのは無駄ってもんです。それにこのほうがソースの視認性がいいと思います（まぁ人それぞれですが）。

BGデータエリアの初期設定は，プログラムの先頭であらかじめ設定してあるX_INITに従って画面を生成しています。いくらか表示画面外まで余計に設定していますが，完全主義者の諸君は必要最低限だけに直したほうが気分がいいでしょう。このプログラムの場合直さなくても問題はありません。

109行目からの割り込みの設定はセオリー通りです。IOCSでV-BLANK割り込みを設定したときになんらかの原因で割り込みを設定できなかった場合，そのままプログラムを終了するようになっています。前回同様，ZP.XなどV-BLANK割り込みを勝手に使用する常駐物には気をつけてください。

119行目からがメイン処理です。ここは1/60ごとにループしますが，このループのことをディスプレイの表示期間に同期するという意味でSYNC（シンク）と呼ぶことにします（特にこの場合，垂直帰線に同期するのでV-SYNCと呼ぶ。水平帰線に同期する場合は当然H-SYNC）。

そこから139行目までは毎SYNC行う動作を記述しています。細かいことはプログラムのコメントを見ればわかりますが，V-BLANKの判定リセットや現在のスクロール速度の表示などです。これが無駄だと思う方はアルゴリズム自体を見直したほうがよいと思います。泥沼のようなプログラムを作る気ならもっと最適化できます。

143〜175行目までは押されたキーによって処理の内容を変えています。上下が押された場合はスクロール速度の変化を，左右が押された場合は，画面の右側を書き換えるか，左側を書き換えるかを選択しています。これはアルゴリズムで説明したとおりですね。

その直後から189行目までがいちばん最初に説明したX軸の読み書きアドレス算出コードです。これも説明したとおりですが，計算後，190，191行で先ほどのキーによって選択された数値を足しています。

193行目からは現在スクロールしているX座標にあわせてスプライトの移動を行っているだけです。半端なコーティングですが，変なことをしてサンプルのプログラムを長くしてもしようがないでしょう。

そして202行目からはESCが押されていたら終了。押されなかったらはじめまでループというだけの処理です。

207行以降の終了処理は説明がなくてもいいでしょう。ただ，割り込みを休止させるときは一応安全のためにCPUの割り込みマスクをかけておきましょう。

さて，219行目からが重要な割り込み処理です。画像関係のI/Oアクセスはすべてこの中でやっています。特にスプライト関連のI/Oは表示をONにしておくとCPUとCRTCのI/Oポートの取り合いでCPU側にウェイトが入ってしまうのです。それではスプライトレジスタに高速にアクセスできないので，通常はCPUがアクセスするときにスプライト関連の表示をOFFにしてしまいます。それをCRTCの表示期間中にOFFにしてしまうと，その処理中だけぽっかり表示されないという感じになってしまいます。それだったらということで，ハード的に画面が表示されていないV-BLANK中にスプライトI/OをOFFにして一気に転送してしまうのが定石となっています。まれに（というほど珍しくないが），この期間中にPCG書き換えなど大量のデータを転送していると，BLANKが終わっても転送が終わらずに表示期間中にはみ出ることがあります。

このプログラムでも座標の移動やデータ転送のスタートアドレスを表示期間中に計算しておき，BLANK中にI/Oに転送しています。

このプログラムではあまり処理の負担にならないということで，毎回BGデータエリアをアクセスしていますが，気になる人はカウンタかなにかを設けてアクセスを必要最低限にするのもよいでしょう。

なんか，改造する価値ありまくりプログラムですが，そこはお勉強ということでいろいろ試してみてください。ちょっとだけ改造のポイントを挙げたいと思います。

◆ 改造のポイント

今回のプログラムは8ドット移動するごとにY軸1ライン分32キャラを書き換えるという処理を行っていますが，よくよく考えるともっといい方法があるのに気づくと思います。

たとえば，Y軸1ラインのキャラクタ数

が32個なのですから，８ドット移動するごとにすべてを書き換えるのではなく，１ドット移動ごとに32/8＝４（または16/8＝2）キャラずつ書き換えたほうが処理が分散してうまくいきそうな感じがします（市販のゲームではやっている）。2ドット単位で移動するときは８または４キャラずつ書き換えるわけです。スクロールスピードが一定の場合はこのほうがよいでしょう。これは課題としておきますが，自分のゲームにマッピング処理を入れる場合はぜひそっちのアルゴリズムで組んでみてください。

やり方はそうは難しくなく，Ｘ座標の下位３ビットまたは４ビットを，書き換えるＹ軸のキャラクタ座標に当てはめればいいのです。あ，いってることがよくわかんないかな。とにかくやってみればいってることがわかります（といって今回も逃げよう）。

その他，このプログラムの縦スクロール版も作ってみてください。たぶん横スクロールよりコーディングが簡単だと思います。横スクロールの場合ビットマップのアクセス方法をよく知らないと難しいんじゃないでしょうか。

◆ 明日に向けてGO! GO! GO!

さて，冒頭で申したとおり今回のサンプルは簡略版です。横スクロールオンリーですし，プログラムも性能より見やすさに重点をおきました。誤解しちゃあ困りますが，私が楽をしようと思ったわけでは（あまり）ありません。

いくらか基本が説明できたところで次回は，

「BGのマッピング処理　応用版」

をやりたいと思います。今度は気合を入れていきますので覚悟してください。もし，ついてこれない人がいたら（私の説明が悪かったら），自作ゲームに組み込めるようなルーチンを予定していますので安心してください。予定だとかなりの初心者にも有効に使える仕様になっています。一応８方向スクロール対応です。もし，制作のノリが悪かったら，ほかのこと（う～んそうだなぁ，パレット制御なんかいいな。いっそのこと未完成のスプライト256システムでも公表しちゃおうかな）にあっさり変更するかもしれません。

それと同時にいくつかのツールを制作したいと思っています。現在考案中なのが，

・BG/マップエディタ
・巨大スプライト制作管理エディタ
・ラスタースクロールデータ生成エディタ

などです。

まだまだゲーム作りに便利なツールというのがあると思うのですが，私の泥色の脳味噌では思いつきません。「俺ならこんなアイデアがあるぜぇ」とか「私のぉ～脳はぁ～世界一ぃぃぃ」などという人はお手紙ください。参考にしたいと思います。

希望としてはSM.Xと統合した環境などいいなぁと思っています。

話ばかり大きくなってしまったような気がしますが，のんびりやっていくつもりなので一歩一歩前に進みましょう。

・協力　H.C.S.

リスト1

```
  1: *
  2: *
  3: *   スーパー平凡なBGのマッピング処理
  4: *
  5: *   カーソル ← → = スクロール        ↑↓ = スピード変化
  6: *                                              ESC = 終了
  7: *
  8: *
  9:     .include        iocscall.mac
 10:     .include        doscall.mac
 11:     .include        const.h
 12:     .include        fefunc.h
 13:
 14: SP_PALET   equ      $e82220              * スプライトパレット
 15: VDC_R1              equ      $e82500 * ビデオコントローラレジスタ1
 16: VDC_R2              equ      $e82600 * ビデオコントローラレジスタ2
 17: SSR                 equ      $eb0000 * スプライトスクロールレジスタ
 18: BSR0_XPOS  equ      $eb0800              * BGスクロールレジスタ0ペー
ジX座標
 19: BG_CTRL             equ      $eb0808 * BGコントローラー
 20: BG0_VRAM   equ      $ebc000              * BG0データエリア
 21:
 22: MAP_ADR             set      $c00000 * グラフィックVRAMを使用
 23: X_INIT              set      000     * マップのX軸の初期位置
 24: MAP_XMAX   set      512                  * マップのX軸の最大値
 25: SPEED_ACS  set      $2000                * スクロールスピードの変化速度
 26:
 27: DI macro
 28:     ori.w   #$0700,sr          * 割り込み禁止
 29:     .endm
 30: EI macro
 31:     andi.w  #$f8ff,sr          * 割り込み許可
 32:     .endm
 33: SPOFF          macro
 34:     move.w  #%00000000_00000000,BG_CTRL
 35:     .endm
 36: SPON           macro
 37:     move.w  #%00000010_00011001,BG_CTRL
 38:     .endm
 39:
 40:     .text
 41:     .even
 42: *
 43: *                       イニシャライズ
 44: *
 45:
 46: init:          moveq.l #$c,d1   * 512*512*65535*1
 47:     IOCS    _CRTMOD          * 画面初期化
 48:     IOCS    _G_CLR_ON        *
 49:     IOCS    _OS_CUROF        * カーソルオフ
 50:     sub.l   a1,a1
 51:     IOCS    _B_SUPER         * スーパーバイザモードへ移行
 52:
 53:     move.w  #%0010000111100100,VDC_R1 * 画面優先順位の設定
 54:
 55: *
 56: *                       マップデータの作成
 57: *                               (グラフィック面にデータを作成する)
 58: map_gene:
 59:     lea.l   mapsym,a1
 60:     IOCS    _SYMBOL
 61: *
 62: *                       速度メッセージ表示
 63: *
 64: m_set00:move.w       #13,d1
 65:     move.w  #30,d2
 66:     IOCS    _B_LOCATE
 67:     pea.l   speedfont
 68:     DOS     _PRINT
 69:     addq.l  #4,sp
 70: *
 71: *                       BGの初期画面の設定
 72: *
 73:     IOCS    _SP_INIT
 74:     IOCS    _SP_ON
 75:     move.w  #%01100001,VDC_R2
 76:     SPOFF
 77:
 78: pcg_set:
 79:     move.l  #0,d1                  * PCGNo. 0をクリア
 80:     IOCS    _SP_CGCLR
 81:     move.l  #$01,d1                * スプライト用PCGをセット
 82:     move.l  #1,d2
 83:     lea.l   pcgdata2,a1
 84:     IOCS    _SP_DEFCG
 85:     move.l  #$ff,d1                * BG用PCGをセット
 86:     move.l  #1,d2
 87:     lea.l   pcgdata1,a1
 88:     IOCS    _SP_DEFCG
 89:
 90:     clr.w   SP_PALET                   * パレット1ー0クリア (黒)
 91:     move.w  coldata+0,SP_PALET+2       * パレット1ー1設定  (青)
 92:     move.w  coldata+2,SP_PALET+4       * パレット1ー2設定  (黄)
 93:     move.w  coldata+4,SP_PALET+6       * パレット1ー3設定  (緑)
 94:
 95:     lea.l   MAP_ADR,a0
 96:     lea.l   BG0_VRAM,a1
 97:     move.w  #X_INIT,d0             * MAPのX座標初期位置補正
 98:     add.w   d0,d0
 99:     andi.l  #MAP_XMAX*2-1,d0
100:     add.l   d0,a0
101:     move.w  #32-1,d0
102: bg_set0:move.w       #64-1,d1
103: bg_set1:move.w       (a0)+,(a1)+
104:     dbra    d1,bg_set1
105:     add.l   #512*2-64*2,a0
```

```
106:     dbra     d0,bg_set0
107:     SPON
108:
109: *
110: *                        割り込み設定
111: *
112:     moveq.l #1,d1             * V-BLANK割り込み設定
113:     lea.l    vblankint,a1
114:     IOCS     _VDISPST
115:     tst.l    d0
116:     bne      quit
117:
118:     EI                        * 割り込み許可
119: *
120: *                        メイン処理
121: *
122: mainloop:
123:     tst.b    b_blank          * V_BLANKの判定
124:     beq      mainloop         * メイン処理は1SYNC中1回しかやらない
125:     move.b   #0,b_blank       * 判定リセット
126:
127: moji_set:                     * スクロール速度の表示
128:     clr.l    d0
129:     move.w   scrl_speed,d0
130:     andi.w   #$0f,d0
131:     lea.l    speedcode,a0
132:     FPACK    __HTOS
133: m_set01:move.w      #41,d1
134:     move.w   #30,d2
135:     IOCS     _B_LOCATE
136:     pea.l    speedcode
137:     DOS      _PRINT
138:     addq.l   #4,sp
139:
140:     move.w   #7,d1
141:     IOCS     _BITSNS
142:
143: udcmp:      btst     #4,d0    * カーソル上
144:     bne      speed_up
145:     btst     #6,d0            * カーソル下
146:     bne      speed_dw
147:
148:     move.w   #7,d1
149:     IOCS     _BITSNS
150: lrcmp:      btst     #3,d0    * カーソル左
151:     bne      map_lft
152:     btst     #5,d0            * カーソル右
153:     bne      map_rit
154:
155:     bra      mainquit         * 何も押されなかった
156: speed_up:
157:     add.l    #SPEED_ACS,scrl_speed
158:     bra      lrcmp
159: speed_dw:
160:     sub.l    #SPEED_ACS,scrl_speed
161:     bra      lrcmp
162:
163: map_lft:
164:     move.w   scrl_speed,d1
165:     andi.w   #$f,d1
166:     sub.w    d1,bg_xscrl      * 左にスクロールする
167:     move.l   #-1,d2           * 画面の右側を書き替える
168:     bra      map_setadr
169: map_rit:
170:     move.w   scrl_speed,d1
171:     andi.w   #$f,d1
172:     add.w    d1,bg_xscrl      * 右にスクロールする
173:     move.l   #33,d2           * 画面の左側を書き替える
174:     bra      map_setadr       * <= 本当はいらない
175: map_setadr:
176:     lea.l    MAP_ADR,a0
177:     lea.l    BG0_VRAM,a1
178: x_ans:
179:     clr.l    d1
180:     move.w   bg_xscrl,d1
181:     asr.l    #4,d1            * 転送X座標を求める
182:     move.w   d1,d3
183:     add.l    d2,d1
184:     asl.l    #1,d1
185:     andi.l   #(512-1)*2,d1
186:     add.l    d1,a0            * 転送する座標 (アドレス)
187:     andi.l   #(64-1)*2,d1
188:     add.l    d1,a1            * 転送される座標 (アドレス)
189:
190:     move.l   a0,mapget_adr
191:     move.l   a1,bgput_adr
192:
193: sp_point:                     * スプライトの座標計算
194:     andi.w   #511,d3
195:     add.w    #16,d3
196:     move.w   d3,sp_scrdat+8*0
197:     move.w   d3,sp_scrdat+8*2
198:     add.w    #16,d3
199:     move.w   d3,sp_scrdat+8*1
200:     move.w   d3,sp_scrdat+8*3
201:
202: mainquit:
203:     move.w   #0,d1
204:     IOCS     _BITSNS               * ESCキーが押されるまでループ
205:     cmp.b    #%0000_0010,d0
206:     bne      mainloop
207: *
208: *                        終了処理
209: *
210: break:      DI
211:     suba.l   a1,a1
212:     IOCS     _VDISPST
213: quit:       EI                     * 割り込みを再許可
214:     IOCS     _OS_CURON             * カーソルオン
215:     moveq.l  #16,d1                * 768*512*16*1=31
216:     IOCS     _CRTMOD               * 画面初期化
217:     IOCS     _G_CLR_ON             *
218:     DOS      _EXIT
219: *
220: *                        割り込み処理
221: *
222: vblankint:
223:     movem.l  d0/a0-a1,-(sp)
224:     move.b   #1,b_blank            * 判定セット
225:     SPOFF
226:     move.l   mapget_adr,a0
227:     move.l   bgput_adr,a1
228:     move.w   #32-1,d0
229: bgline_set:
230:     move.w   (a0),(a1)
231:     adda.l   #512*2,a0
232:     adda.l   #64*2,a1
233:     dbra     d0,bgline_set
234:
235:     lea.l    sp_scrdat,a0
236:     lea.l    SSR,a1
237:     move.w   #4*2*4/4-1,d0
238: spatr_set:
239:     move.l   (a0)+,(a1)+
240:     dbra     d0,spatr_set
241:
242:     move.w   bg_xscrl,BSR0_XPOS
243:
244:     SPON
245:     movem.l (sp)+,d0/a0-a1
246:     rte
247: ********************************************
248:     .data
249:     .even
250: pcgdata1:   * BG用PCG
251:     .dc.l   $22222222,$21111111,$21111111,$21111111
252:     .dc.l   $21111111,$21111111,$21111111,$21111111
253:     .dc.l   $21111111,$21111111,$21111111,$21111111
254:     .dc.l   $21111111,$21111111,$21111111,$22222222
255:     .dc.l   $22222222,$11111112,$11111112,$11111112
256:     .dc.l   $11111112,$11111112,$11111112,$11111112
257:     .dc.l   $11111112,$11111112,$11111112,$11111112
258:     .dc.l   $11111112,$11111112,$11111112,$22222222
259: pcgdata2:   * スプライト用PCG
260:     .dc.l   $33333333,$30000000,$30000000,$30000000
261:     .dc.l   $30000000,$30000000,$30000000,$30000000
262:     .dc.l   $30000000,$30000000,$30000000,$30000000
263:     .dc.l   $30000000,$30000000,$30000000,$30000000
264:     .dc.l   $33333333,$00000000,$00000000,$00000000
265:     .dc.l   $00000000,$00000000,$00000000,$00000000
266:     .dc.l   $00000000,$00000000,$00000000,$00000000
267:     .dc.l   $00000000,$00000000,$00000000,$00000000
268:
269: coldata:    * BGのパレットに設定するカラー  (G_R_B_I)
270:     .dc.w   %00000_00000_11111_0    * 青
271:     .dc.w   %11111_11111_00000_0    * 黄
272:     .dc.w   %11111_00000_00000_0    * 緑
273: mapsym:     .dc.w   0,4
274:     .dc.l   mapfont
275:     .dc.b   1,1
276:     .dc.w   $01ff
277:     .dc.b   2,0
278: mapfont:    * BGのマップに使用するテストパターン
279:     .dc.b   "春夏秋冬晴曇雨雪霧雹虹頭胴手足指爪髪毛垢",00
280:     .even
281: speedcode:
282:     .ds.b   4
283: speedfont:
284:     .dc.b   "現在のスクロールドット数は$00です。",00
285:     .even
286: mapget_adr:
287:     .dc.l   MAP_ADR                 * 転送するマップのファーストアドレス
288: bgput_adr:
289:     .dc.l   BG0_VRAM                * 転送されるBGのファーストアドレス
290: bg_xscrl:
291:     .dc.w   X_INIT                  * スクロール値
292: sp_scrdat:                          * スプライトの転送データ
293:     .dc.w   16,16,%00_00_0001_00000001,%011
294:     .dc.w   32,16,%01_00_0001_00000001,%011
295:     .dc.w   16,32,%10_00_0001_00000001,%011
296:     .dc.w   32,32,%11_00_0001_00000001,%011
297: scrl_speed:
298:     .dc.w   $04,$00                 * スクロールスピード
299: b_blank:                            * B_BLANK判定ワーク
300:     .ds.b   1
```

第142部 S-OSで学ぶZ80マシン語講座(4)

●応募してくれてありがとう

このまま，希望者のいないまま終わってしまうかもしれないと危惧していた「YGCSver.0.20モニタ募集」。ついに，呼びかけに応じてくれた人が出ました。う〜む，めでたい。さっそく，システム一式を送らせていただきました。じっくり使ってみてください。

モニタ募集は引き続き行いますので，興味のある方は，遠慮なくハガキをお送りください。待ってます。

なお，ダンプリストの掲載はもう少し先になります。そのときには，2月号で紹介したものより多少バージョンアップしてしまうかもしれませんが，簡単なサンプルとともに掲載してきちんとフォローしていくつもりです。

●Z80マシン語プログラミング

コマンドも徐々に追加され，ようやくディスクエディタらしくなりつつある「Z80マシン語講座」。完成まであとわずか，伊藤氏には最後までがんばってもらいたいですね。

さて，S-OSを使うことでZ80マシン語を学んでしまおうというこの連載。当初の「個々の命令よりも，どのように命令を組み合わせてプログラミングをしていくかに重点を置く」という言葉どおり，実際にプログラムを作り，それを解説してきました。

最終的にこの連載から，Z80でのプログラミング，そしてS-OSでのプログラミング作法を学んでもらえれば，連載の趣旨が達成されたことになるのですが……どうでしょうか？

まあ，

S-OSを使う＝サービスコールを利用

とひと口にいってしまえるくらいですから，S-OSを使うこと自体は難しくありません。基本的に必要な引数をレジスタにセットして，目的のサービスコールを呼び出すだけです。キャラクタの1文字を表示したければ，

```
LD      A,"A"
CALL    #PRINT
```

だけで済みますしね。

しかし，今回取り上げた題材がディスクエディタということもあり，S-OSを使うのは難しいんじゃないか，複雑な知識が必要なのでは？　と思い込んでしまった人，そんなことはありません。

確かに，ファイルの内部構造まで言及しているため，知識のない人にとって理解するのが難しいところもあります。理解できないからといってあきらめていたのでは進歩がありません。

そこで，こういった理論を理解するときのコツとして，とりあえず納得してしまう，という方法があります。連載では，あとあとその鵜呑みにしていた知識に対する別の角度からの解説が必ずあります。そのときに再びとりあえず納得していた知識と照らし合わせ，理解を深めていけばいいのです。

```
D0:07 C9 D6 30 D8 FE 0A 38 07 FE 11 D8 D6 07 FE 10
E0:3F C9 C5 1A 13 CD D2 02 38 0D 0F 0F 0F 0F 4F 1A
F0:13 CD D2 02 38 01 B1 C1 C9 CD E2 02 67 D4 E2 02
=1
=R21
Record 0021
   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
00:20 81 83 81 83 81 83 20 82 72 81 7C 82 6E 82 72  / <<< S-OS
10:20 82 94 82 95 82 92 82 82 82 8F 20 20 20 82 72  / turbo   S
20:82 76 82 6E 82 71 82 63 20 81 84 81 84 81 84 20  /WORD >>>
30:0D 00 2A 7E 1F E9 CD 4F 05 CD DD 01 0C 20 3C 3C
40:3C 20 20 53 2D 4F 53 74 75 72 62 6F 20 20 53 57  /< S-OSturbo  SW
50:4F 52 44 20 3E 3E 3E 0D 00 C3 32 01 00 00 00 00  /ORD >>>
60:00 00 00 00 00 00 00 00 00 00 00 00 21 20 21 C9
70:F5 C5 3E 20 4F C3 91 01 F5 C5 06 0D 48 AF C3 9C
80:01 F5 3A 95 01 B7 C2 79 01 F1 C9 F5 C5 06 19 4F
90:0A 47 FE 0D 3E 14 C2 9B 01 3E FF 3C 32 95 01 3A
A0:7C 1F B7 79 C4 F9 01 78 01 91 17 DF C1 F1 C9 00
B0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F5
C0:D5 1A FE 0D 28 14 CD 8B 01 13 C3 C1 01 F5 D5 1A
D0:B7 28 07 CD 8B 01 13 C3 CF 01 D1 F1 C9 E3 7E 23
E0:B7 28 06 CD 8B 01 C3 DE 01 E3 C9 3A 95 01 90 3F
F0:D8 CD 70 01 3C C2 F1 01 C9 C5 F5 3A 30 1C B7 C2
=
DA:    DB:    DE:    HL    A    DIR:    TYPE:    STAT:    NAME:
```

もちろん，間違って理解していたのなら即座に訂正しましょう。あとは，これを繰り返していくだけです。

あまりいばれる方法ではないし，時間も手間もかかりますが，最終的に正しい知識が身につけばいいのです。結果よければすべてよし。気負ってくじけるより，C調に，お気楽に構えてがんばりましょう。

●S-OS "SWORD" とは？

誌面がX68000に染まるにつれてどんどん異彩を放つTHE SENTINEL，というわけでもないのかもしれませんが，やはり新しい読者の中にはこのS-OS "SWORD" を知らない人がちらほら出てきています。

皆さんご承知のとおり，現在のコンピュータはメーカー，機種によって互換性がありません。もちろん，8ビット機全盛の頃も状況は同じでした。そこで当時のユーザーは考えました。機種は違えども同じZ80というCPUを使っているのだから，なんとか同じソフトを使えるようにならないか，と。

そこで，まずはシャープ製のマシンのBIOSレベルで共通化を図ったのです。さらには，Z80を搭載した他社製のマシンにまで侵食し，さらにZ80エミュレーションプログラムにより，FM-7,PC-9801,X68000までも巻き込み，勢力を伸ばしていきました。

つまり，BIOSのエントリルーチンの共通化を行ったのがS-OSであり，そのためS-OSシステムが搭載されている機種なら，このTHE SENTINELで掲載されているソフト資産が共有できるのです。

最近では活動が地味になってしまっていますが，まだまだがんばりますよ。

1994■インデックス

全機種共通
S-OS"SWORD"要

S-OSで学ぶ Z80マシン語講座(4)

Itou Masahiko
伊藤 雅彦

今月もどんどんコマンドを追加していきます。前回までに学んだ，ディスクのファイル構造をきっちり把握しておけば大丈夫。理解できたら自分の手で，オリジナルコマンドを作ってみよう。

寒い日が続くなか，今月もやってまいりましたマシン語講座です。私，寒いのは大っ嫌いなんですが，部屋の中でパソコンをやっていれば北風も関係ありません。寒さに負けまいなんて思わず，寒さとの勝負を放棄してぬくぬくと講座を始めましょう。

Dコマンド

では，さっそくDコマンドを作りましょう(図1)。初回(1993年11月号)に掲載したコマンド一覧では仕様がわかりにくかったかもしれません（使用例が間違ってるし)。ここで補足説明をしましょう。Dコマンドは3とおりの使い方があって，まず，

=D

で全ディレクトリデータを表示します。

=D3

以上のようにファイル番号を指定すると，そのファイルのディレクトリデータを表示します。

=D3 T8000

このように変更項目まで指定すれば，ディレクトリデータの書き換えになります。この例ではファイル番号3のファイルの先頭アドレスが8000_Hに書き換えられます。

Dコマンドのルーチンを作るうえでいちばんややこしそうなのが，このようにパラメータの数によって処理内容が変わるところです。パラメータの解釈をしながら，どの処理をするのかを判断しなくてはいけません。

では，実際にパラメータをどう解釈していけばいいのか考えてみましょう。

まず，ファイル番号の指定があるかどうかを確認します。ここは16進数の値で指定されるところですからPARAMETERルーチンをコールして，Zフラグが1なら全ディレクトリデータの表示だと判断すればいいわけですね。

ファイル番号があったなら，今度は変更項目指定があるか調べることになるんですが，これにはSPCUTルーチンを使います。つまり，ファイル番号のあとのスペースを読み飛ばして，次のパラメータの文字を見て00_Hだったらもうパラメータはないということです。したがって，指定ファイルのディレクトリデータ表示の処理に移ります。パラメータがあった場合には，データの書き換え処理ということになります。

これを整理すると，

1) PARAMETERルーチンをコール
　Z＝1→全データ表示処理
　Cy＝1→エラー

2) SPCUTルーチンをコール
　A＝00_H→指定データ表示処理
　その他→データ書き換え処理

となります。

さて，全データの表示処理と指定データの表示処理では，プログラムをかなり共通化できそうです。指定データ表示ルーチンを作っておいて，全データの表示処理のときには指定ファイル番号を1から順にカウントしながら指定データ表示ルーチンをコールすれば，全部のデータが表示されることになりますからね。

このことを踏まえて作ったサブルーチンがDCOMS1です。Cレジスタに入っている内部ファイル番号のディレクトリデータを表示します。内部ファイル番号というのは，ファイル番号から1を引いたものです。通常，ファイル番号は1から始まることになっていますが，プログラム内部では0から始まっていたほうがなにかと扱いやすいので，内部ファイル番号というものを導入しました。

データ書き換えの場合の処理では，書き換えるデータの種類ごとにプログラムを用意しましたが(変更項目指定が"S"，"T"，"E"ならDCOM6以降，"A"ならDCOM7以降，"N"ならDCOM8以降)，処理内容は似たようなものです。ディレクトリをディスクからメモリに読み込み，メモリ上で書き換えて，ディスクに書き込むという作業をしているだけです。

ここではレジスタの値を保護するためにEXX命令を使っています。レジスタの値の保護にはPUSH，POP命令を使うのが一般的ですが，EXXのほうがPUSH，POPより高速ですから，EXXが使えるときにはこちらを選びたいものです。でも，実際には使えるときというのがなかなかないもので，最初のうちはいらぬバグを呼ばないために，「レジスタ保護にはPUSH，POPを使う」と決めておいたほうが無難でしょう。

Cコマンド

次はCコマンドを作ります。これはファイルがどのクラスタに記録してあるかを表示するものです（図2)。

また表示だけではなく変更もできるようにします。ただし，変更というのはあくまでもFATに書いてあるクラスタ連鎖を書き換えるだけであって，クラスタの内容そのものを別のクラスタに移動するわけではありません。つまり，ファイルの中身が変わってしまうことになるわけです。わけも

わからず変更すると確実にファイル内容を破壊してしまいます。この変更機能はあくまでも壊れたディスクの修復用と心得ておいてください。

Cコマンドの機能は，Dコマンドと同様に3つあります。

=C

これは全ファイルのクラスタ連鎖の表示です。次には，

=C3

のようにファイル番号を指定すれば，そのファイルのクラスタ連鎖を表示します。

=C3 19 1A 1B 1F 85

とすると，クラスタ連鎖の変更になります。この例では，ファイル番号3のファイルのクラスタ連鎖を19_H→$1A_H$→$1B_H$→$1F_H$として，最後の第$1F_H$クラスタの使用セクタ数を6としています。つまり，クラスタ番号を順に指定したあとに，最終クラスタの使用セクタ数を($7F_H$+使用セクタ数)で指定するということです。

このコマンドもDコマンドと似たようなもので，ディレクトリの表示がクラスタ連鎖の表示に変わっただけですね。だからプログラムのおおまかな作りはDコマンドからいただいてしまえばいいんです。

まず，パラメータから処理内容を判断する方法ですが，Dコマンドと違う点は，Cコマンドでは2番目以降のパラメータも16進数だということです。そこのところを踏まえると，判断の手順は，

1) PARAMETERルーチンをコール
  Z=1→全データ表示処理
  Cy=1→エラー
2) PARAMETERルーチンをコール
  Z=1→指定データ表示処理
  Cy=1→エラー
  その他→データ書き換え処理

となります。

全ファイルの表示と指定ファイルの表示とでプログラムを共通化できる点もDコマンドと同様です。指定ファイルのクラスタ連鎖を表示する共通サブルーチンを作ればどちらの場合でも使えるわけです。

問題はどうやってクラスタ連鎖を表示するかです。ここはDコマンドと違いますからね(同じだったら，それはまるっきりのDコマンドです)。どうすればいいかというと，まずディレクトリを見て，31バイト目の先頭クラスタ番号を表示します。

次にFATに目を移して，先頭クラスタ番号をnとすると，(FATの先頭+n)のアドレスのデータを読みます。そのデータが80_H以上なら続きのクラスタはないということですから，表示が終わりです。$7F_H$以下なら，それが次に続くクラスタのクラスタ番号ですから表示します。

そして，そのクラスタ番号をnとして再び(FATの先頭+n)のアドレスのデータを読む処理を繰り返していきます。基本的にはこれだけのことです。わかるでしょうか？ わからなかった方は連載初回のディスク管理方法の説明を読み返してください。

クラスタ連鎖の表示を実際にやっているのが，CCOMS1というサブルーチンです。最初にファイル番号，ファイル名，先頭クラスタ番号を表示します。そして，そのファイルが消去済みのファイルだったら，クラスタ連鎖を表示せずに処理を終了します。なぜかといえば，もう消されてしまったファイルは，FATにクラスタ連鎖の情報が残っていないからです。

その次が本題のクラスタ連鎖表示になるんですが，ここでは連鎖がループしていないかチェックしながら表示しています。普通は連鎖がループすることはありませんが，FATが壊れていることも考えられます。先ほどいった表示手順では，連鎖がループしていると処理が永久に終わらなくなってリセットをかけるしかなくなってしまいます。最悪の場合でもコマンド入力に戻ってくるようでないと，やはりまずいですよね。

どうやってチェックしているかというと，まずCOMWKという名前で128バイトのワークエリアをとり，最初にすべて2を入れておきます。そして第nクラスタが使用クラスタだとわかると，(COMWK+n)のアドレスのデータを-1します。これをクラスタ番号の表示処理と並行して行っていきます。そうすると，連鎖がループしている場合には一度表示，減算処理したクラスタをもう一度処理しようとすることになりますから，そのときにCOMWKのデータの減算結果が0になります(図3)。これをチェックすればいいわけです。

さて，クラスタ連鎖の変更の場合の処理も考えてみましょう。これには，ディレクトリの先頭クラスタ情報とFATの2つを書き換える操作が必要です。ラベルCCOM5以降のプログラムがそれですが，やっていることは，

1) 2番目以降のパラメータの値をCOMWKに移す
2) ディレクトリとFATをディスクから読む
3) 処理を行うファイルの旧クラスタ連鎖をFATから消去する
4) ディレクトリの先頭クラスタ情報を書き換える
5) FATにクラスタ連鎖を書き込む
6) ディレクトリとFATをディスクに書き込む

です。なんのテクニックもないプログラムになっていますから，コメントを頼りに自分で解析してみてください。プログラムコードを見ながらフローチャートを書いてみ

今月使用した システムサブルーチン&ワークエリア

●サブルーチン

#FILE：Aレジスタにファイル属性，DEレジスタペアにファイル名が入っている先頭アドレスをセットしてコールすると，(#IBFAD)に属性とファイル名，(#DSK)にデバイス名をセット。

#DWTSB：セクタの書き込み。(#DSK)にデバイス，DEレジスタペアに書き込み開始レコード番号，Aレジスタに書き込みセクタ数，HLレジスタペアに書き込み元アドレスをセットしてコール。

●ワークエリア

#IBFAD：インフォメーションブロックの先頭アドレス。インフォメーションブロックのデータ構造はディレクトリと同じ。

図1 Dコマンドの表示形式

```
03  0*  FILE1      .OBJ  1A61  3000  3000
(1) (2)     (3)           (4)   (5)   (6)
```

(1) ファイル番号
(2) ファイル属性(*印は書き込み禁止属性)
(3) ファイル名
(4) ファイル長
(5) ロード先頭アドレス
(6) 実行アドレス

図2 Cコマンドの表示形式

```
04  FILE2      .OBJ  09-0A-21  :  8C
(1)     (2)            (3)         (4)
```

(1) ファイル番号
(2) ファイル名
(3) クラスタ連鎖
(4) 最終クラスタの使用セクタ数(+$7F_H$)

れば理解できるでしょう。

ちなみに，1)の処理でCOMWKをパラメータ値の退避場所として使っていますが，このワークはクラスタ連鎖の表示処理では別の使い方をしていましたね。使用メモリを節約するために，同じワークをいろいろな用途に使っているわけなんです。

Oコマンド

S-OSのモニタでDコマンドを使うとディレクトリが表示されますが，表示されるファイルの順番を変えたいときに役に立つのがOコマンドです。たとえば，

Asc file1.txt:0000：12A3：0000
Asc file2.txt:0000：20BB：0000
Asc file3.txt:0000：1AF2：0000
Asc file4.txt:0000：3918：0000

と並んでいるものを，

Asc file3.txt:0000：1AF2：0000
Asc file1.txt:0000：12A3：0000
Asc file2.txt:0000：20BB：0000
Asc file4.txt:0000：3918：0000

と並べ替えたいときには，

=O1 2 4

とパラメータ指定します。1～2番目のファイルを4番目のファイルの前に移動するという意味になります。また，

=O3 1

でも同じことができます。この場合は，3番目のファイルを1番目のファイルの前，つまり先頭に移動するという意味になります。このように1つのファイルだけを先頭に移動する場合は，

=O3

だけでもOKです。

負の数の扱い

リストのラベルOCOMIの直後を見てください。BCの値から32を引くためにADD命令を使っています。HLレジスタに−32を入れてBCに足しているわけですね。ここで，HLレジスタに−32を入れるというのはどういうことでしょう。ADD命令に負の数を入れることができるんでしょうか。

実は−32とは16進数でFFE0$_H$のことなんです。8000$_H$～FFFF$_H$は−32768～−1とみなして扱えるんです。たとえば，

32−32=0020$_H$+FFE0$_H$=0000$_H$=0
10−1=000A$_H$+FFFF$_H$=0009$_H$=9
−4−3=FFFC$_H$+FFFD$_H$=FFF9$_H$=−7

というふうに。ただしこれは2バイト値の場合で，1バイト値なら80$_H$～FF$_H$を−128～−1として扱えます。なお，正負を反転するには，「ビット反転して1を足す」という操作をします。たとえば100なら，

100=0064$_H$=0000 0000 0110 0100$_B$
↓ビット反転
1111 1111 1001 1011$_B$
↓+1
−100=FF9C$_H$=1111 1111 1001 1100$_B$

という具合に−100になります。Aレジスタの1バイト値の正負を反転するNEG命令もチェックしておいてくださいね。

図3 COMWKの働き（Cコマンド表示処理時）

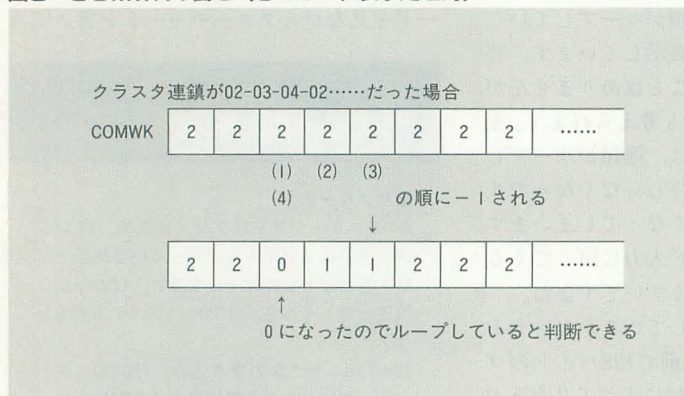



図4 ディレクトリのガーベジコレクション

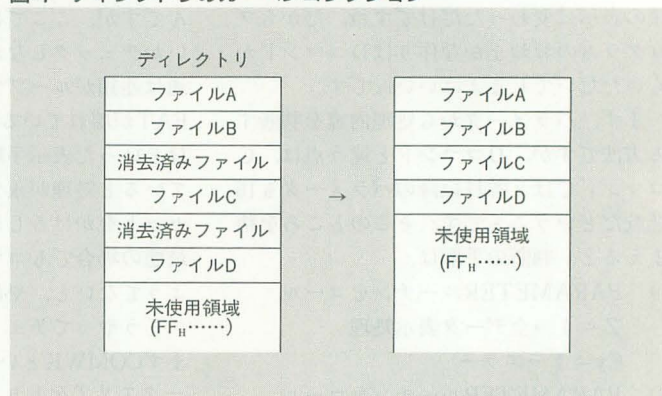
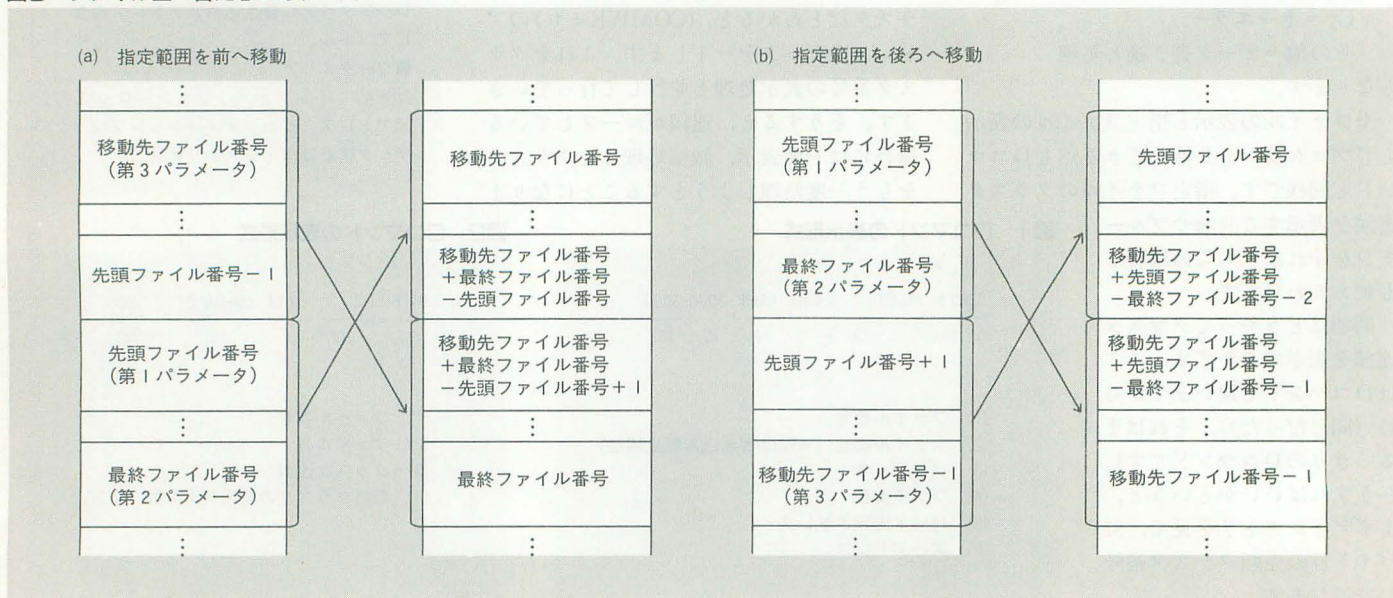



図5 ファイル並べ替え2つのケース

そして，パラメータなしで，

=O

とすると，ディレクトリ領域のガーベジコレクションをします。これはどういうことかというと，消去済みファイルのディレクトリデータ（ファイル属性が00_Hのもの）はいらないからなくしてしまって，データを前に詰めよう，ということです（図4）。

と，こういう仕様のものを作りたいわけです。まずはガーベジコレクションの処理から片づけましょう。これは簡単で，ディレクトリの頭から順にファイル属性を調べていって，00_Hなら後方のデータを前に詰めていけばいいんです。

OCOMルーチンのラベルOCOM3の手前までがそのプログラムなんですが，処理内容が単純なわりにはわかりにくいものになってしまったようです。DEレジスタにディレクトリの注目位置のアドレスが入っていて，BCレジスタには注目位置より後ろのデータのバイト数が入っていることを頭に入れて，プログラムの動作を追ってください。レジスタにどんな値をもたせるかというのが，プログラムを作るときのひとつのポイントなんですよね。

それでは，ディレクトリの並べ替えにいきましょう。ラベルOCOM3以降を見てください。

まず問題になるのがパラメータの処理です。パラメータが1個，2個，3個の場合があるわけですが，これはすべてパラメータ3個の形に直すことができます。

つまり，

=Oa b　→　=Oa a b

=Oa　　→　=Oa a 1

というわけです。このような変換をしてパラメータの値をB，C，Lレジスタにセットする（ただし内部ファイル番号の形で）のが最初にしている処理です。このように一般化すれば，どの場合でも同じように処理できて楽になるでしょう。

で，さらに一般化したいことがあるんです。並べ替えのとき，指定した範囲を前に移動する場合と後ろに移動する場合とがありえるわけです。これをどちらかの形に統一しておかないとあとの処理が面倒になってしまいます。そんなわけで，後ろに移動する場合には，

B=最終内部ファイル番号+1

C=移動先内部ファイル番号-1

L=先頭内部ファイル番号

として，前に移動する形に直すことにします（図5）。

そのあと，ディレクトリを読み込んで並べ替えをするんですが，その手順は，

1)　内部ファイル番号がL～B-1のファイルのディレクトリデータをCOMWKに退避

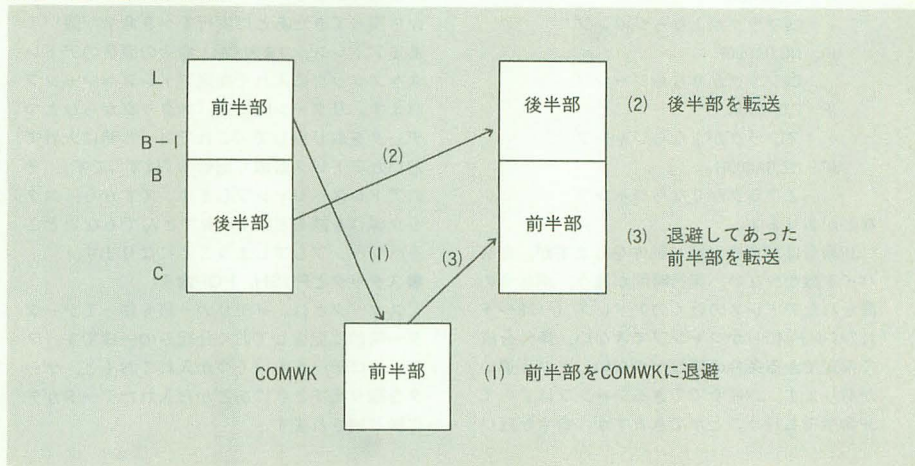
2)　内部ファイル番号がB～Cのファイルのディレクトリデータを，内部ファイル番号L以降の位置に転送

3)　COMWKに退避したデータを2)の転送先最終アドレス以後に転送

となっています（図6）。リストではラベルOCOM7以降になりますが，このあたりはいろいろなデータがレジスタのあちこちを飛び回って読みにくくなっています。

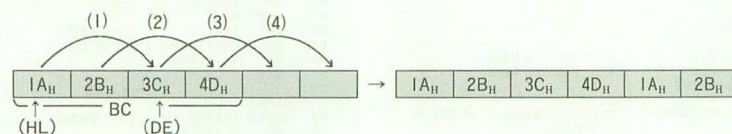
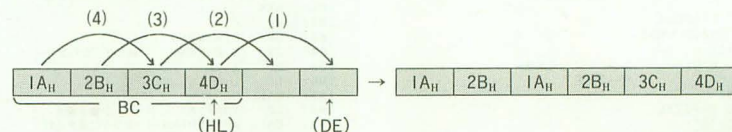
Z80のようにレジスタが少なくて演算機能が弱いCPUでは，プログラムを最適化するとすぐにこんなコードが出来上がる可能性があります。あまり最適化すると，作った本人でもよく理解できないプログラムになることもよくあります。初心者の方はそういうことにはあまりこだわらずに，とにかく「動く」プログラムを作ることを目指してください。実際，本当に骨の髄まで最適化された速度が必要とされるプログラムというのは，そうないはずです。

今月はこれでおしまいです。ではまた。

図6　並べ替えの手順

ブロック転送

一定の領域のデータを別の場所に転送するときに使う命令といえば，もちろんLDIR命令です。HLレジスタに転送元の領域の先頭アドレス，DEレジスタに転送先の領域の先頭アドレス，BCレジスタに転送領域のバイト数を入れてLDIR命令を実行すればいいんですね。

でも，転送先と転送元の領域が重なっていた場合はどうでしょう。後方の領域に転送しようとすると，LDIR命令ではうまくいきません。アドレスの小さいほうのデータから順に転送されるので，後ろのほうのデータは転送される前に新しいデータで上書きされてしまいます。このようなときにはLDDR命令を使います。HLレジスタに転送元の領域の最終アドレス，DEレジスタに転送先の領域の最終アドレス，BCレジスタに転送領域のバイト数を入れてLDDR命令を実行するだけです。

前方の領域への転送ならLDIR命令，後方の領域への転送ならLDDR命令，とうまく使い分けてください。

Z80基礎知識(3)

●JP，JR命令

BASICでいうところのGOTO文みたいなものです。普通，命令は上から下へ（というかアドレスの小さいほうから大きいほうへ）順番に実行されていきますが，JP，JR命令で任意のアドレスに実行を移すことができます。

```
JP  09200H
```

以上の命令でアドレス9200Hに置かれている命令に実行が移ります。

また，フラグの状態が条件を満たしたときだけジャンプを行う条件分岐命令として，

```
JP  C,0A000H
      Cyフラグが1ならジャンプ
JP  NC,0A000H
      Cyフラグが0ならジャンプ
JP  Z,0A000H
      Zフラグが1ならジャンプ
JP  NZ,0A000H
      Zフラグが0ならジャンプ
```

などがあります。

JR命令はJP命令と同じ動作をしますが，命令バイト数が少ない，実行時間が違う，JR命令の置かれたアドレスの近くのアドレス（-128～+127バイト）にしかジャンプできない，条件分岐で指定できる条件の種類が少ない，という違いがあります。JR命令でできるジャンプはすべてJP命令でも行うことができますが，命令が短いことから，JR命令が使える場合にはできるだけ使ってやりたいところです。

●CALL，RET命令

CALLはサブルーチンをコールする命令，RET命令はサブルーチンからリターンする命令です。BASICのGOSUB，RETURNみたいなものです。JP命令と同様に，

```
CALL  NC,0CC00H
RET  Z
```

というような条件をつけることができます。

コール，リターンの処理はスタックを用いて実現しています。コール時には，サブルーチンから戻ってきたあとに実行すべき命令が置いてあるアドレス，つまりCALL命令の直後のアドレスをスタックに入れて指定アドレスへジャンプします。リターン時には，スタックからひとつデータを取り出して（これでコール時に入れておいたアドレスが取り出せる“はず”です），そのアドレスへジャンプします。ですから，スタック操作を誤るとRET命令でとんでもないところへジャンプしてしまうことになります。

●スタックとPUSH，POP命令

スタックとは，メモリの一部を使ってデータを一時的に記憶しておく仕組みの一種です。スタックにデータをいくつか入れておくと，データを取り出すときにあとから入れたデータが先に取り出されます。

Z80では出し入れするデータは2バイトと決まっています。データを入れる命令はPUSH，取り出す命令はPOPです。

```
PUSH  HL
PUSH  DE
POP  BC
POP  DE
```

以上のようなプログラムでは，DEレジスタの値がBCレジスタに，HLレジスタの値がDEレジスタに入る結果になります。

スタックの動作を制御するためのレジスタがSP（スタックポインタ）で，常に次に取り出されるべきデータの入っているアドレスを値としてもっています。PUSHのときにはSPの値を2減らして，その値のアドレスにデータを書き，POPのときにはSPの値のアドレスのデータを読み，SPの値を2増やすという動作が行われます。

●DJNZ命令

このDJNZ命令は，

```
DEC B
JR  NZ,xxxxH
```

を1命令でやってしまうもので，Bレジスタをカウンタにしてループ処理をするときに便利です。カウンタループを作るときには，Bレジスタをカウンタにできないか常に考えたいところです。

リスト

```
  1: ;
  2: ; ラベル定義
  3: ;
  4: #FILE:  EQU 01FA3H
  5: #DWTSB: EQU 02003H
  6: ;
  7: #IBFAD: EQU 01F74H
  8: ;
  9: ; コマンド処理ルーチン
 10: ;
 11: ;
 12: ; D Command
 13: ;
 14: DCOM:
 15:  CALL PARAMETER
 16:  JP   C,#BELL
 17:  JR   NZ,DCOM3   ; 第1パラメータがあれば DCOM3 へ
 18: ;                * 全データ表示
 19:  CALL READDIR
 20:  RET  C
 21:  LD   C,0
 22: DCOM1:
 23:  CALL DCOMS1     ; 1行表示
 24:  RET  C
 25:  CALL #PAUSE
 26:  DW   DCOM2
 27:  INC  C
 28:  JR   DCOM1
 29: DCOM2:
 30:  RET
 31: ;
 32: DCOM3:
 33:  DEC  L
 34:  RES  7,L        ; L の bit7 をリセット
 35:  LD   C,L        ; C = 内部ファイル番号
 36:  CALL SPCUT
 37:  OR   A
 38:  JR   NZ,DCOM5   ; 第2パラメータがあれば DCOM5 へ
 39: ;                * 指定データ表示
 40:  EXX             ; C を保護
 41:  CALL READDIR
 42:  RET  C
 43:  EXX
 44:  CALL CNTDIR     ; A = ファイル数
 45:  CP   C
 46:  JR   C,DCOM4
 47:  JP   NZ,DCOMS1
 48: DCOM4:
 49:  JP   BADFNO     ; A<=C ならエラー
 50: ;                * データ変更
 51: DCOM5:
 52:  CALL CAPITAL
 53:  CP   'N'
 54:  JR   Z,DCOM8
 55:  CP   'A'
 56:  JR   Z,DCOM7
 57:  LD   B,18
 58:  CP   'S'
 59:  JR   Z,DCOM6
 60:  LD   B,20
 61:  CP   'T'
 62:  JR   Z,DCOM6
 63:  CP   'E'
 64:  JP   NZ,#BELL
 65:  LD   B,22
 66: ;                * ファイル長、先頭アドレス、実行アドレス
 67: DCOM6:
 68:  CALL PARAMETER ; HL = 変更データ値
 69:  JP   C,#BELL
 70:  JP   Z,#BELL
 71:  EXX
 72:  CALL READDIR
 73:  RET  C
 74:  CALL CNTDIR     ; A = ファイル数
 75:  EXX
 76:  CP   C
 77:  JP   C,BADFNO
 78:  JP   Z,BADFNO   ; A<=C ならエラー
 79:  PUSH HL         ; 変更データ値保存
 80:  LD   L,C
 81:  CALL DIRADR
 82:  LD   E,B
 83:  LD   D,0
 84:  ADD  HL,DE
 85:  POP  DE
 86:  LD   (HL),E
 87:  INC  HL
 88:  LD   (HL),D     ; データ書き換え
 89:  CALL WRITEDIRS ; セクタに書き込む
 90:  JR   NC,DCOMS1
 91:  RET
 92: ;                * ファイル属性
 93: DCOM7:
 94:  CALL PARAMETER ; L = 変更属性データ
 95:  JP   C,#BELL
 96:  JP   Z,#BELL
 97:  EXX
 98:  CALL READDIR
 99:  RET  C
100:  CALL CNTDIR     ; A = ファイル数
101:  EXX
102:  CP   C
103:  JP   C,BADFNO
104:  JP   Z,BADFNO   ; A<=C ならエラー
105:  LD   A,L
106:  LD   L,C
107:  CALL DIRADR
108:  LD   (HL),A     ; データ書き換え
109:  CALL WRITEDIRS ; セクタに書き込む
110:  JR   NC,DCOMS1
111:  RET
112: ;                * ファイル名
113: DCOM8:
114:  EXX
115:  LD   DE,(KBPTR)
116:  LD   A,1
117:  CALL #FILE      ; インフォメーションブロック
118:                  ; にファイル名が入る
119:  CALL READDIR
120:  RET  C
121:  CALL CNTDIR     ; A = ファイル数
122:  EXX
123:  CP   C
124:  JP   C,BADFNO
125:  JP   Z,BADFNO   ; A<=C ならエラー
126:  LD   L,C
127:  CALL DIRADR
128:  INC  HL
129:  EX   DE,HL
130:  LD   HL,(#IBFAD)
131:  INC  HL
132:  LD   A,C
133:  LD   BC,16
134:  LDIR            ; データ書き換え
135:  LD   C,A
136:  CALL WRITEDIRS ; セクタに書き込む
137:  RET  C
138: ;JR   DCOMS1
139: ;
140: ;  in ---- C  = 内部ファイル番号
141: ;  out --- Cy = 属性が FFH (1)／ FFH 以外(0)
142: ;  break - F, A, B, DE, HL
143: ;
144: DCOMS1:
145:  CALL PRTRSET
146:  LD   L,C
147:  CALL DIRADR
148:  LD   A,0FEH
149:  CP   (HL)
150:  RET  C          ; 属性が FFH なら RET
151: ;
152:  LD   A,C
153:  INC  A          ; A = ファイル番号
154:  CALL #PRTHX
155: ;
156:  CALL #PRNTS
157: ;
158:  LD   A,(HL)     ; A = 属性
159:  OR   A
160:  JR   NZ,DCOMS11
161:  LD   A,'K'
162:  CALL #PRINT
163:  CALL #PRNTS
164:  JR   DCOMS14
165: DCOMS11:
166:  LD   D,A
167:  AND  0BFH       ; bit6 をリセット
168:  LD   E,'O'
```


▶良質のAVGをやりたいのですが，なにかお勧めありませんか？　星　武史(19)山形県

```
169:  CP    1              ; Bin 属性か
170:  JR    Z,DCOMS12
171:  LD    E,'B'
172:  CP    2              ; Bas 属性か
173:  JR    Z,DCOMS12
174:  LD    E,'A'
175:  CP    4              ; Asc 属性か
176:  JR    Z,DCOMS12
177:  LD    A,D
178:  CALL  #PRTHX
179:  JR    DCOMS14
180: DCOMS12:
181:  LD    A,E
182:  CALL  #PRINT
183:  LD    A,' '
184:  BIT   6,D            ; 書き込み禁止属性か
185:  JR    Z,DCOMS13
186:  LD    A,'*'
187: DCOMS13:
188:  CALL  #PRINT
189: ;
190: DCOMS14:
191:  CALL  #PRNTS
192:  INC   HL
193:  CALL  PRTFN
194:  INC   HL
195: ;
196:  LD    B,3
197: DCOMS15:
198:  CALL  #PRNTS
199:  LD    E,(HL)
200:  INC   HL
201:  LD    D,(HL)
202:  INC   HL
203:  EX    DE,HL
204:  CALL  #PRTHL
205:  EX    DE,HL
206:  DJNZ  DCOMS15
207: ;
208:  CALL  #LTNL
209:  OR    A              ; Cy=0
210:  RET
211: ;
212: ; C Command
213: ;
214: CCOM:
215:  CALL  PARAMETER
216:  JP    C,#BELL
217:  JR    NZ,CCOM3
218: ;                          * 全データ表示
219:  CALL  READDIR
220:  CALL  NC,READFAT
221:  RET   C
222:  LD    C,0
223: CCOM1:
224:  CALL  CCOMS1
225:  RET   C
226:  CALL  #PAUSE
227:  DW    CCOM2
228:  INC   C
229:  JR    CCOM1
230: CCOM2:
231:  RET
232: ;
233: CCOM3:
234:  DEC   L
235:  RES   7,L
236:  LD    C,L            ; C = 内部ファイル番号
237:  CALL  PARAMETER
238:  JP    C,#BELL
239:  JR    NZ,CCOM5
240: ;                          * 指定データ表示
241:  EXX
242:  CALL  READDIR
243:  CALL  NC,READFAT
244:  RET   C
245:  EXX
246:  CALL  CNTDIR         ; A = ファイル数
247:  CP    C
248:  JR    C,CCOM4
249:  JP    NZ,CCOMS1      ; A<=C ならエラー
250: CCOM4:
251:  JP    BADFNO
252: ;                          * 連鎖書き換え
253: CCOM5:
254:  LD    A,L
255:  DEC   A
256:  CP    07FH
257:  JP    NC,#BELL       ; L が 01H~7FH でなければエラー
258:  LD    A,C
259:  EX    AF,AF'         ; 第1パラメータ保護
260:  LD    BC,COMWK
261: CCOM6:
262:  LD    A,L
263:  DEC   A
264:  CP    08FH
265:  JP    NC,#BELL       ; L が 01H~8FH でなければエラー
266: CCOM7:
267:  LD    A,L
268:  LD    (BC),A
269:  INC   BC
270:  CALL  PARAMETER
271:  JP    C,#BELL
272:  JR    NZ,CCOM6
273: ;
274:  XOR   A
275:  LD    (BC),A         ; エンドマーク (00H) を書く
276:  EX    AF,AF'
277:  LD    C,A
278:  EXX
279:  CALL  READDIR
280:  CALL  NC,READFAT
281:  RET   C
282:  CALL  MKFATUSE
283:  CALL  CNTDIR         ; A = ファイル数
284:  EXX
285:  CP    C
286:  JP    C,BADFNO
287:  JP    Z,BADFNO       ; A<=C ならエラー
288: ;
289:  CALL  KILLFAT
290: ;
291:  LD    L,C
292:  CALL  DIRADR
293:  LD    B,(HL)         ; B = 属性
294:  LD    DE,30
295:  ADD   HL,DE
296:  LD    DE,COMWK
297:  LD    A,(DE)
298:  LD    (HL),A         ; 先頭クラスタ情報を変更
299:  INC   B
300:  DEC   B              ; B=0 なら Z=1
301:  JR    Z,CCOM11       ; 消去済ファイルなら処理終了
302: ;
303: CCOM8:
304:  LD    HL,(#FATBF)
305:  ADD   A,L
306:  LD    L,A
307:  JR    NC,CCOM9
308:  INC   H              ; HL = HL + A
309: CCOM9:
310:  LD    A,(HL)
311:  OR    A
312:  JR    NZ,CCOM10      ; 使用中クラスタへ連鎖していたら
                                              CCOM10 へ
313:  INC   DE
314:  LD    A,(DE)
315:  OR    A
316:  JR    Z,CCOM12       ; 使用セクタ数の指定がなければエラー
317:  LD    (HL),A
318:  CP    080H
319:  JR    C,CCOM8
320: CCOM10:
321:  INC   DE
322:  LD    A,(DE)
323:  OR    A
324:  JR    NZ,CCOM12      ; 未処理パラメータが残っていたらエラー
325: ;
326:  CALL  WRITEFAT
327:  RET   C
328: CCOM11:
329:  CALL  WRITEDIRS
330:  JR    NC,CCOMS1
331:  RET
332: ;
333: CCOM12:
334:  CALL  #MPRNT
335:  DM    'Bad Chain Data'
336:  DB    00DH,0
337:  JP    #BELL
338: ;
339: ;   in ---- C  = 内部ファイル番号
340: ;   out --- Cy = 属性が FFH (1) / FFH 以外 (0)
341: ;   break - F, A, B, DE, HL, BC', DE', HL'
342: ;
343: CCOMS1:
344:  CALL  PRTRSET
345:  LD    L,C
346:  CALL  DIRADR
347:  LD    A,0FEH
348:  CP    (HL)
349:  RET   C              ; 属性が FFH なら RET
350: ;
351:  LD    A,C
352:  INC   A
353:  CALL  #PRTHX
354:  CALL  #PRNTS
355:  LD    D,(HL)         ; D = 属性
356:  INC   HL
357:  CALL  PRTFN
358:  CALL  #PRNTS
359: ;
360:  LD    A,D
361:  LD    DE,13
362:  ADD   HL,DE
363:  LD    D,A
364:  LD    A,(HL)
365:  LD    E,A
366:  CALL  #PRTHX
367: ;
368:  INC   D
369:  DEC   D
370:  JR    Z,CCOMS14      ; 消去済なら表示終了
371: ;
372:  LD    A,E
373:  ADD   A,A
374:  JR    C,CCOMS14      ; E > 7FH なら表示終了
375: ;
376:  EXX                  ; 先頭クラスタ番号 (=E) 保護
377:  LD    HL,COMWK
378:  LD    DE,COMWK+1
379:  LD    BC,127
380:  LD    (HL),2
381:  LDIR
382:  EXX
383: ;
384:  LD    D,0
385:  JR    CCOMS12
386: CCOMS11:
387:  LD    E,A
388:  LD    A,'-'
389:  CALL  #PRINT
390:  LD    A,E
391:  CALL  #PRTHX
392: CCOMS12:
393:  LD    HL,COMWK
394:  ADD   HL,DE
395:  DEC   (HL)           ; ループチェック
396:  JR    Z,CCOMS13
397:  LD    HL,(#FATBF)
398:  ADD   HL,DE
399:  LD    A,(HL)
400:  CP    080H
401:  JR    C,CCOMS11
402: ;
403:  LD    E,A
404:  LD    A,':'
405:  CALL  #PRINT
406:  LD    A,E
407:  CALL  #PRTHX
408:  JR    CCOMS14
409: ;
410: CCOMS13:
411:  CALL  #MPRNT
412:  DM    ':Loop'
413:  DB    0
414: ;
415: CCOMS14:
416:  CALL  #LTNL
417:  OR    A              ; Cy=0
418:  RET
419: ;
420: ; O Command
421: ;
422: OCOM:
423:  CALL  PARAMETER
424:  JP    C,#BELL
425:  JR    NZ,OCOM3
426: ;                          * ガーベジコレクション
427:  CALL  #MPRNT
428:  DM    'Garbage Collection'
429:  DB    00DH,0
430:  CALL  READDIR
431:  RET   C
432:  CALL  CNTDIR
433:  OR    A
434:  RET   Z
435:  LD    L,A
436:  EX    AF,AF'         ; ファイル数保護
437:  INC   L
438:  LD    H,0
439:  ADD   HL,HL
440:  ADD   HL,HL
441:  ADD   HL,HL
442:  ADD   HL,HL
443:  ADD   HL,HL
444:  LD    C,L
445:  LD    B,H            ; BC = 後方のデータのバイト数 + 32
446:  LD    DE,DIRBF
447: ;
448:  JR    OCOM2
449: OCOM1:
450:  LD    HL,-32
451:  ADD   HL,BC
452:  LD    C,L
453:  LD    B,H            ; BC = BC - 32
454:  LD    HL,32
455:  ADD   HL,DE
456:  EX    DE,HL          ; DE = DE + 32 ,
                                              HL = 旧 DE 値
457:  DEC   A
458:  JR    NZ,OCOM2       ; 消去済でなければ OCOM2 へ
459: ;
460:  EX    DE,HL
461:  PUSH  DE
462:  PUSH  BC
463:  LDIR
464:  POP   BC
465:  POP   DE
466: ;
467: OCOM2:
468:  LD    A,(DE)         ; A = 属性
469:  INC   A              ; A=FFH なら Z=1
470:  JR    NZ,OCOM1
471: ;
472:  EX    AF,AF'
473:  DEC   A
474:  LD    B,A
475:  XOR   A
476:  JP    WRITEDIRP
477: ;                          * 並べ替え
478: OCOM3:
479:  DEC   L
480:  RES   7,L
481:  LD    B,L            ; B = 第1パラメータ
482:  CALL  PARAMETER
483:  JP    C,#BELL
484:  JR    NZ,OCOM4
485: ;
486:  LD    C,B
487:  LD    L,0
488:  JR    OCOM6
489: ;
490: OCOM4:
491:  DEC   L
492:  RES   7,L
493:  LD    C,L            ; C = 第2パラメータ
494:  CALL  PARAMETER
495:  JP    C,#BELL
496:  JR    NZ,OCOM5
497: ;
498:  LD    L,C
499:  LD    C,B
500:  JR    OCOM6
501: ;
502: OCOM5:
503:  DEC   L
504:  RES   7,L            ; L = 第3パラメータ
505:  LD    A,C
506:  CP    B
507:  JP    C,#BELL
508: ;
509: OCOM6:
510:  LD    A,L
511:  CP    B
512:  JR    C,OCOM7        ; 前方への移動なら OCOM7 へ
513:  RET   Z
514:  INC   C
515:  CP    C
516:  RET   Z
517:  JP    C,#BELL
518:  LD    L,B
519:  LD    B,C
520:  LD    C,A
521:  DEC   C
522: ;
523: OCOM7:
524:  EXX
525:  CALL  READDIR
526:  RET   C
527:  CALL  CNTDIR
528:  EXX
529:  LD    H,A
530:  LD    A,C
531:  CP    H
532:  JP    NC,BADFNO
533: ;
534:  LD    H,C
535:  PUSH  HL             ; 並べ替え範囲保護
536: ;
537:  LD    A,C
538:  SUB   B
```

▶スピタル産業のVRマウスはメチャクチャ快適だぞ。　　伴 武士(23)千葉県

```
539:  INC  A
540:  LD   C,A          ; C = 後部ファイル数
541:  LD   A,B
542:  SUB  L            ; A = 前部ファイル数
543:  CALL DIRADR
544: ;
545:  EX   DE,HL
546:  LD   L,A
547:  LD   H,0
548:  ADD  HL,HL
549:  ADD  HL,HL
550:  ADD  HL,HL
551:  ADD  HL,HL
552:  ADD  HL,HL        ; HL = A * 32
553:  LD   A,C
554:  LD   C,L
555:  LD   B,H
556:  EX   DE,HL
557:  LD   DE,COMWK
558:  PUSH BC           ; 前部データバイト数保護
559:  PUSH HL           ; 前部データ先頭アドレス保護
560:  LDIR              ; 命令実行後、HL は転送元最終
                                   アドレス + 1
561:                    ; つまり後部データ先頭アドレス
                                   になっている
562: ;
563:  EX   DE,HL
564:  LD   L,A
565:  LD   H,0
566:  ADD  HL,HL
567:  ADD  HL,HL
568:  ADD  HL,HL
569:  ADD  HL,HL
570:  ADD  HL,HL        ; HL = A * 32
571:  LD   C,L
572:  LD   B,H
573:  EX   DE,HL
574:  POP  DE
575:  LDIR              ; 命令実行後、DE は転送先
                              最終アドレス + 1
576:                    ; つまり前部データ移動先
                              アドレスになっている
577: ;
578:  POP  BC
579:  LD   HL,COMWK
580:  LDIR
581: ;
582:  POP  BC
583:  LD   A,C
584:  JP   WRITEDIRP
585: ;
586: ; 基本サブルーチン群
587: ;
588: ;==================== Dコマンド分
589: ;
590: ; PRTFN
591: ;    ファイル名表示
592: ;
593: ;  in ---- HL = ファイル名先頭アドレス
594: ;  out --- HL = HL + 16
595: ;  break - F, A, B
596: ;
597: PRTFN:
598:  LD   B,13
599: PRTFN1:
600:  LD   A,(HL)
601:  CALL #PRINT
602:  INC  HL
603:  DJNZ PRTFN1
604:  LD   A,'.'
605:  CALL #PRINT
606:  LD   B,3
607: PRTFN2:
608:  LD   A,(HL)
609:  CALL #PRINT
610:  INC  HL
611:  DJNZ PRTFN2
612:  RET
613: ;
614: ; WRITEDIRS
615: ;    ディレクトリ書き込み (C が記録されているセクタのみ)
616: ;
617: ;  in ---- C  = 内部ファイル番号
618: ;  out --- Cy = 書き込み失敗(1)／成功(0)
619: ;  break - F, A, BC (Cy=1 の時), DE, HL,
                                       F',A'
620: ;
621: WRITEDIRS:
622:  LD   A,C
623:  AND  0F8H
624:  LD   L,A
625:  CALL DIRADR      ; HL = 書き込みデータ先頭アドレス
626:  LD   DE,(#DIRPS)
627:  RRCA
628:  RRCA
629:  RRCA             ; A = 内部ファイル番号 / 8
630:  ADD  A,E
631:  LD   E,A
632:  JR   NC,WRITEDIRS1
633:  INC  D            ; DE = DE + A
                          (書き込みレコード番号)
634: WRITEDIRS1:
635:  LD   A,(DEVICE)
636:  LD   (#DSK),A
637:  LD   A,1
638:  CALL #DWTSB
639:  RET  NC
640:  CALL #ERROR
641:  SCF
642:  RET
643: ;
644: ; DIRADR
645: ;    ファイル L のディレクトリアドレス計算
646: ;
647: ;  in ---- L  = 内部ファイル番号
648: ;  out --- HL = DIRBF 内先頭アドレス
649: ;  break - F, DE
650: ;
651: DIRADR:
652:  LD   H,0
653:  ADD  HL,HL
654:  ADD  HL,HL
655:  ADD  HL,HL
656:  ADD  HL,HL
657:  ADD  HL,HL
658:  LD   DE,DIRBF
659:  ADD  HL,DE
660:  RET
661: ;
662: ; CNTDIR
663: ;    ディレクトリ登録ファイル数計算
664: ;
665: ;  out --- A = ディレクトリ登録数
666: ;  break - F, B, DE, HL
667: ;
668: CNTDIR:
669:  LD   HL,DIRBF-32
670:  LD   DE,32
671:  LD   A,-1
672: CNTDIR1:
673:  INC  A
674:  ADD  HL,DE
675:  LD   B,(HL)
676:  INC  B            ; B=FFH なら Z=1
677:  JR   NZ,CNTDIR1
678:  RET
679: ;
680: ; BADFNO
681: ;    ファイル番号エラー表示
682: ;
683: ;  break - F, A, DE
684: ;
685: BADFNO:
686:  CALL #MPRNT
687:  DM   'Bad File Number'
688:  DB   00DH,0
689:  JP   #BELL
690: ;==================== Cコマンド分
691: ;
692: ; WRITEFAT
693: ;    FAT 書き込み
694: ;
695: ;  out --- Cy = 書き込み失敗(1)／成功(0)
696: ;  break - F, A, BC (Cy=1 の時)
                                , DE, HL, F',A'
697: ;
698: WRITEFAT:
699:  LD   A,(DEVICE)
700:  LD   (#DSK),A
701:  LD   DE,(#FATPOS)
702:  LD   HL,(#FATBF)
703:  LD   A,1
704:  CALL #DWTSB
705:  RET  NC
706:  CALL #ERROR
707:  SCF
708:  RET
709: ;
710: ; KILLFAT
711: ;    FAT 上でのファイル消去
712: ;
713: ;  in ---- C = 内部ファイル番号
714: ;  break - F, A, B, DE, HL
715: ;
716: KILLFAT:
717:  LD   A,C
718:  INC  A
719:  LD   HL,FATUSE
720:  LD   DE,(#FATBF)
721:  LD   B,128
722: KILLFAT1:
723:  CP   (HL)
724:  JR   NZ,KILLFAT2
725:  EX   DE,HL
726:  LD   (HL),0
727:  EX   DE,HL
728: KILLFAT2:
729:  INC  HL
730:  INC  DE
731:  DJNZ KILLFAT1
732:  RET
733: ;==================== Oコマンド分
734: ;
735: ; WRITEDIRP
736: ;    ディレクトリ書き込み (A～B が記録されているセクタのみ)
737: ;
738: ;  in ---- A = 先頭内部ファイル番号
739: ;          B = 最終内部ファイル番号
740: ;  out --- Cy = 書き込み失敗(1)／成功(0)
741: ;  break - F, A, BC, DE, HL, F',A'
742: ;
743: WRITEDIRP:
744:  AND  0F8H
745:  LD   L,A
746:  CALL DIRADR
747:  LD   DE,(#DIRPS)
748:  RRCA
749:  RRCA
750:  RRCA
751:  LD   C,A          ; C = 先頭内部ファイル番号 / 8
752:  ADD  A,E
753:  LD   E,A
754:  JR   NC,WRITEDIRP1
755:  INC  D
756: WRITEDIRP1:
757:  LD   A,(DEVICE)
758:  LD   (#DSK),A
759:  LD   A,B
760:  AND  0F8H
761:  RRCA
762:  RRCA
763:  RRCA             ; A = 最終内部ファイル番号 / 8
764:  SUB  C
765:  INC  A
766:  CALL #DWTSB
767:  RET  NC
768:  CALL #ERROR
769:  SCF
770:  RET
771: ;
772: ; ワークエリア
773: ;
774: COMWK:
775:  DS 4096
```

▶ 全 機 種 共 通 シ ス テ ム イ ン デ ッ ク ス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

▶やはりOh!Xは受験雑誌だった。センター試験で大爆発！　　河合　雅弘(18)長崎県

第22部　magiFORTH TRACER
第23部　ディスクダンプ＆エディタ
第24部　“SWORD”2000 QD
連載　対話で学ぶmagiFORTH
特別付録　PC-8801版S-OS“SWORD”
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS“SWORD”
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版S-OS“SWORD”
■86年9月号
第28部　FuzzyBASIC発表
連載　明日に向かってmagiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタDREAM
第31部　FuzzyBASIC料理法<1>
■86年11月号
第32部　パズルゲームHOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC料理法<2>
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC料理法<3>

1987

■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC料理法<4>
■87年2月号
第36部　アドベンチャーゲームMARMALADE
第37部　テキアベ作成ツールCONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツールMAGE
付録　“SWORD”再掲載とMAGICの標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS“SWORD”変身セット
第43部　MZ-700用“SWORD”をQD対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASICコンパイラ
第45部　エディタアセンブラZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージJACKWRITE
特別付録　FM-7/77版S-OS“SWORD”
■87年9月号
第49部　リロケータブル逆アセンブラInside-R
特別付録　PC-8001/8801版S-OS“SWORD”
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASICコンパイラの拡張
第52部　X1turbo版S-OS“SWORD”
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OSの仲間たち
第53部　もうひとつのFuzzyBASIC入門
第54部　ファイルアロケータ＆ローダ
インタラプト　S-OSこちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo版“SWORD”アフターケア
　　　　ラインプリントルーチン
特別付録　PASOPIA7版S-OS“SWORD”

1988

■88年1月号
第58部　FuzzyBASICコンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲームELFES
■88年3月号
第60部　構造型コンパイラ言語SLANG
■88年4月号
第61部　デバッギングツールTRADE
第62部　シミュレーションウォーゲームWALRUS
■88年5月号
第63部　シューティングゲームELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語SLANG入門(1)
第66部　Lisp-85用NAMPAシミュレーション
■88年7月号
第67部　マルチウィンドウドライバMW-1
連載　構造化言語SLANG入門(2)
■88年8月号
第68部　マルチウィンドウエディタWINER
■88年9月号
第69部　超小型エディタTED-750
第70部　アフターケアWINERの拡張
■88年10月号
第71部　SLANG用ファイル入出力ライブラリ
第72部　シューティングゲームMANKAI
■88年11月号
第73部　シューティングゲームELFES IV
■88年12月号
第74部　ソースジェネレータSOURCERY

1989

■89年1月号
第75部　パズルゲームLAST ONE
第76部　ブロックゲームFLICK
■89年2月号
第77部　高速エディタアセンブラREDA
特別付録　X1版S-OS“SWORD”<再掲載>
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOR
　　　　OBAN
■89年4月号
第79部　SLANG用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータRING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲームTICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲームPUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号
第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ
■90年2月号
第89部　超小型コンパイラTTC++
■90年3月号
第90部　超多機能アセンブラOHM-Z80
■90年4月号
第91部　ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部　インタプリタ言語STACK
■90年6月号
第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲームSQUASH!
第95部　X68000対応S-OS“SWORD”
特別付録　PC-286対応S-OS“SWORD”
■90年7月号
第96部　リロケータブルアセンブラWZD
■90年8月号
第97部　リンカWLK
■90年9月号
第98部　BILLIARDS
■90年10月号
第99部　ライブラリアンWLB
■90年11月号
第100部　タブコード対応エディタEDC-T
■90年12月号
第101部　STACKコンパイラ

1991

■91年1月号
第102部　ブロックアクションゲームCOLUMNS
■91年2月号
第103部　ダイスゲームKISMET
■91年3月号
第104部　アクションゲームMUD BALLIN'
■91年4月号
第105部　SLANG用カードゲームDOBON
■91年5月号
第106部　実数型コンパイラ言語REAL
■91年6月号
第107部　Small-C処理系の移植
■91年7月号
第108部　REALソースリスト編
■91年8月号
第109部　Small-Cライブラリの移植
■91年9月号
第110部　SLANG用NEWファイル出力ライブラリ
■91年10月号
第111部　Small-C活用講座（初級編）
■91年11月号
第112部　Small-C活用講座（応用編）
第113部　MORTAL
■91年12月号
第114部　Small-C SLANGコンパチ関数

1992

■92年1月号
第115部　LINER
■92年2月号
第116部　シミュレーションゲームPOLANYI
■92年3月号
第117部　カードゲームKLONDIKE
■92年4月号
第118部　オプティマイザO80実践Small-C講座(1)
■92年5月号
第119部　COMMAND.OBJ実践Small-C講座(2)
■92年6月号
第120部　COMMAND.OBJ2実践Small-C講座(3)
■92年7月号
第121部　関数リファレンス実践Small-C講座(4)
■92年8月号
第122部　ワイルドカード実践Small-C講座(5)
第123部　グラフィックライブラリ GRAPH.LIB
■92年9月号
第124部　O-EDIT&MODCNV
■92年10月号
第125部　SLENDER HUL実践Small-C講座(6)
■92年11月号
第126部　EDIT実践Small-C講座(7)
■92年12月号
第127部　MAKE実践Small-C講座(8)

1993

■93年1月号
第128部　EDC-Tの拡張
■93年2月号
第129部　BLACK JACK
■93年3月号
第130部　シューティングゲームコアシステム作成法(1)
■93年4月号
第131部　シューティングゲームコアシステム作成法(2)
■93年5月号
第132部　シューティングゲームコアシステム作成法(3)
■93年6月号
第133部　REVERSI
■93年7月号
特別付録　MSX用S-OS“SWORD”
■93年8月号
第134部　MACINTO-C再掲載
■93年9月号
第135部　7並べ
特別付録　SLANG再々掲載
■93年10月号
第136部　シューティングゲームコアシステム作成法(4)
■93年11月号
第137部　S-OSで学ぶZ80マシン語講座(1)
■93年12月号
第138部　エディタアセンブラREDA再掲載

▶最近（よい意味で）Xシリーズに嵐の前の静けさのようなものを感じているのですが，真実はどうなのやら。
飯嶋　徹(22)神奈川県

石の言葉，言葉の夢

[新連載]

支援すれども管理せず

Ogikubo Kei
荻窪 圭

ではでは，新春大型放屁ってことで，といっても，もう3月か。月日の経つのは早いものだなあ。

冬になると，人間どうなるか，っていうと，怠慢になる。寒いし鬱陶しいしめんどくさい。だから，いろいろとアイデアが湧く。ボコボコと沸く。沸騰すると熱いから，適当なところで，冷ます。冬だから，すぐに冷める。人肌あたりで取り出して，パソコンに流し込む。すると，こういう原稿ができるわけだ。わはは。

支援すれども管理せず

家でゴロゴロしてて考えるのは，ホームコンピューティングのこと。ちょっと真面目。

ホームコンピューティングっていうと，思い出すのが電脳住宅だ。行ったことはないから半分以上想像で書いているのだが，「電脳住宅」のポイントは，「管理」にある。見えないところにコンピュータがあり，家を，そして生活を管理してくれる。「コンピューター＝管理する機械」。見事に美しい図式だ。が，考えてみよう。

家全体がコンピュータだった。20年前のSFショートショートのオチとどこが違うのか。そのショートショートはこんな話のはずだ。未来，何でも機械がやってくれる快適な家があって，気持ちいいのだが，実はその家自体が機械であり，人々は機械の胎内で遊ばされていただけだった。

なぜ，ホームコンピューティングという言葉から「管理」が出てくるのだろう。オフィスにコンピュータを入れる→省力化，って発想から抜け出ていないからである。家庭にコンピュータを入れる→省力化。家庭内で負担になる仕事→掃除炊事洗濯子守冷暖房戸締まりなどなど→生活の管理，という図式である。はじめから「管理」しようとしているのではなく，従来のオフィスにあるコンピュータのイメージを家庭に持ち込んだら，はからずも，生活を管理するものになってしまった，というわけだ。あらら，である。

これはいろんな面で非現実的だ。

まず，家の電脳住宅化は金がかかる。床をひっぺがし，壁にケーブルをはわせ，窓から風呂場からトイレから，みな電脳化せねばならない。誰がするもんか，ってんで。せいぜい，いまの家を模様替えする程度の，ちょっとリフォームする程度の感覚でできねばウソだ。

続いて，機械に依存する恐怖である。機械は絶対ではない。当たり前だ。バグはどこかにいる。当たり前だ。生活を機械に委ねるとき，バグも一緒に抱え込みたいだろうか。否である。停電の危険，急激な温度変化の危険，何も知らん3歳の子供がダイレンジャーごっこのついでに何しでかすかわかんない危険。何にしろ，無人運転は危険だ，ってなわけで，電脳システムに生活の根幹を押さえられるほどホラーなことはない。ホラーである。スプラッタである。

では，ホームコンピュータはどっから登場するか。ほんとに，家庭にコンピュータが入ったりするのか。

キーポイントは，「支援すれども管理せず」である。「管理」されるほうが好きな人が大勢いるのは重々承知のうえであるが，私は嫌いだ。嫌いだから，そういうシステムは許さないのである。もちろん，セキュリティシステム的なものを否定したりはしない。セキュリティシステムの電脳化は許す。でも，生活を管理されるほど無気味なことはない。年頃の娘を持つ親が娘の生活を管理しようとしているわけではないのだ。管理したからといって，娘がイケイケにならない保証はないのである。やるやつはどう転んでもやる。

で，「管理」ではなく「支援」する。さまざまな生活における行動を支援する。スタンドアロンで十分なものもあれば，ネットワーク化したほうがいいものもある。まずは，家庭内ハイテク機器分野から強化していくのが考えやすい。

家庭内ハイテク機器。といえば。まずは「通信」と「娯楽」である。この2つは欠かせない。続いて，白モノ家電だ（エアコン，冷蔵庫，電子レンジなどなどの白いモノ)。「通信」とか「娯楽」系の機械はみな黒いなあ。黒モノ家電とはいわないのかしら。

白モノ家電については，スタンドアロンで十分なものが多い。これ以上進化すると「支援」から「管理」へ移行してしまうおそれがある。「～しなさい」の嵐になってしまうわけだ。ワンボタンで済む洗濯機，電子レンジ，エアコン，炊飯器。これらがネットワーク化したからといって，コストに見合う成果は得られないだろう。

娯楽の電脳

家庭内通信機器といえば，電話とFAXである。一部，モデムである。電話といってもいろいろあるが，親子式留守電付きコードレスホンくらいが一般的なところ。

家庭内娯楽といえば「放送」か「パッケージメディア」である。放送ってのはテレビやラジオ。パッケージメデ

ィアはCDやビデオやLDやファミコン。おまけとして，ハンディビデオカメラとかカセットテープレコーダといったモノを加える。

最後に，パソコンを加える。

どれも，支援はするが管理はしないモノたちだ。ホームコンピュータが入り込む隙間は，まずこれら「通信&メディア」なのである。なぜなら，パソコンのメディア化という流れの延長線上にあるからだ。

要するに，テレビを中心としたシステムとパソコンの融合。電話を中心としたシステムとパソコンの結合から考えてみよう，ってわけだ。

まずはテレビのほうから。

●AVシステムとCDメディア

現実の話からする。

パソコンがマルチメディア化した。パソコンの側から見ると，「いやあ，パソコンで実写ビデオが見られるとかパソコンが喋るとか，時代は進歩したものですねぇ」ってなるのだが，それはどうでもいい視点である。パソコン側からマルチメディアを見てもしょうがない。それでは，テクノロジーの進歩でフルカラーサンプリング音源高速マイクロプロセッサ万歳，で終わってしまう。パソコン側からではなく，メディア享受者の側から見る。

すると，従来のさまざまなメディアのおいしいところをデジタル化してパソコンならではのデータの制御（インタラクティブとかいわれているやつだ）を加えて，そうすれば，新しいメディアができるじゃないか。それをマルチメディアと呼ぼう，ってことになる。つまりはそういうことだ。

それをパソコン中心に考えると，パソコンのモニタにフルカラー(あるいは16ビットカラー。65536色といってもいい）グラフィック，パソコンのサンプリング音源。メディアはCD-ROM，ってことになる。パソコンユーザーにとってみれば，いままでのパソコンの用途がぐんと広がって面白い。でも，あくまでもそれはパソコンユーザーにとっての話だ。たいていの人はパソコンユーザーではない。だから，部屋の隅に置いたパソコンのモニタで何かコトが起きても，しょうがないのである。

CD-ROMってメディアを考えてみよう。これは，いろいろ欠点はあるものの，あと数年はパッケージの主流になるだろう。

CDメディアにどんなデジタルデータを突っ込むことができるか。

・CD-ROM：パソコンのソフトウェアやデータ
・CD-DA：要するに，音楽CD
・PhotoCD：コダックがはじめた，銀塩フィルムをデジタル化してライトワンスCDに焼く技

さらには，新しいネタとして，

・ビデオCD：MPEG1を使い，CDメディアに映像を記録する技。将来を期待されているが，カラオケメディアで終わったらヤダなあ

ってのがある。けっこうなものだと思う。これに3DOなんかのゲーム機を加えると，音楽・写真・映像・ゲームなどなどがみな同じメディアで供給されるってことなのである。うまく作れば，ひとつの機械でみな再生できる，ってわけなのだ。

となると，パソコン側もそれなりのサービスをしなければならなくなる。音楽CDを気持ちよく聴けるだけのサウンド出力，大画面で見られるだけの映像，ってわけだ。こういったことを追求し始めると，つらい。価格の問題もある。

そうなるとどうしても，AVシステムの一環としてパソコンを位置づけるしかなくなる。25インチとか31インチのテレビへの出力，オーディオシステムへの出力。中心にパソコン。

だが，そんなところにパソコンを置いたら，それまでのパソコンとしての使い方に影響してくる。自分の領域で静かにパソコンを使いたいことだってある。

そこで，「コアシステム」が登場する。なんて，大袈裟なものではないが。

要するに，パソコンとテレビをいかにうまくつなぐか，ってことだ。

コアとなるのは，薄型軽量のノートブックパソコンだ。できたら，2kg以下の軽いやつ。これがコア。単体として使えば，単なるノートブックパソコン。

テレビにはコアとドッキングするためのマルチメディアユニットがつながっている。このユニットはCD-ROMドライブを内蔵しており，上記のCDメディアすべてに対応することが可能だ。さらに，コアにおさまりきらない，フルカラーのビデオ回路とそれをNTSC信号に変換するスキャンコンバータ，サンプリング音源が入っている。それだけ。コアをこのユニットにシステムバスでドッキングする。すると，テレビがモニタになり，オーディオシステムがサウンドモニタになるのだ。いいスキャンコンバータを使えば，文字をなんとか読めるくらいの出力は可能だ。

大画面テレビで遊ぶ。今年は，パソコンとテレビの結合が流行するぞ。私はそう読んでいるのだ。

でもって，コアをつないでないときのマルチメディアユニットは，音楽CDプレイヤー兼PhotoCDプレイヤー兼ビデオCDプレイヤーだ。単体でもちゃんと動くように作っておく。パソコンがつながれば，それがインテリジェントプレイヤーになり，パソコンのソフトも遊べ，PhotoCDは加工でき，音楽CDはサンプリングされるのである。

で，ワープロしたり通信したりゲームしたりしたいときは，パソコン部分だけを持っていって普通のノートブックパソコンとして使う。

この，ノートブックパソコン用マルチメディアシステム。98,000円程度で作れたら，売れるんじゃないかと思う。そうすれば，家庭にあるノートブックパソコンも家族みんなが楽しめる娯楽機械として株も上がるはずだ。

なんていっている間に，誌面が尽きてしまった。だから，電話の話は来月に回す。

では，そういうことで。

そしてマウスは運動不足になる

指しゃぶりとマック

Macintosh(以下マック)がデビューしてからちょうど10年たちます。ごく一部の人が計算するためのものにすぎなかった電子計算機が，今日のようにごくごく自然に家庭のなかで使われるようになったこの変化は，革命的という言葉をつけてもおおげさではないほどの重要な意味をもっていると思います。

飛び抜けたアイデアがてんこ盛りになっているマックは，10年のあいだに次第に人々をとりこにしてきました。さまざまなアイデアのなかでひときわ光るのが，「グラフィカルユーザーインタフェイス」，「デスクトップ環境」，「マウス」などのキーワードで説明のできるオペレーティングシステムであり，操作環境であるといえます。

電子計算機とはこの現実世界とはまったく異なる近寄りがたいブラックボックスなのだ，という人々の気持ちを，マックの画面に映し出されるデスクトップ環境は追っ払ってくれたのでした。画面にはごみ箱だの書類だの時計だのがころがっています。そのような比喩を用いることにより，電子計算機のなかに本当は広がっているはずの暗黒世界をうまく隠蔽してくれたのでした。

図1　3歳児でも簡単に描いた絵

たとえば，同居している3歳の子ども(性別：男)，いまだに指しゃぶりをやめないようですが，PowerBookにすぐに慣れてしまい，なにやらいじって喜んでいるのです。いきなり電源ジャックにスプーンを差してショートさせてしまい，修理に出す羽目になったときにはあせりましたが。

驚くべきことは，特に教えたわけでもないのに，ダブルクリックやドラッグなどの操作を自由にこなしていることです。トラックボールを右に動かせばマウスポインタが画面上で右に動くとか，ボタンを押したままトラックボールを動かすと画面上のものも一緒に動くとかいうことは，そんなに自然なことなのかと首をひねりたい気持ちにもなってきます。

その3歳児がマックを使って描いた絵を1枚載せておきます。ソフトの使い方など特に説明したわけでもなく，自分で作ったものです。「へーっ，やるなー！」などとおだてはしましたが。ちなみに，このソフトウェアはPDS(Public Domain Software)の「Mac Tuberling」という名前のもので，じゃがいもの絵の上に福笑いのように目鼻口などをのせて遊びます。

3歳児でもすぐになじめる最大の理由，それがデスクトップ環境であり，グラフィカルユーザーインタフェイスであり，マウスを使った入力であると思います。しかも，異なるソフトウェアにおいてもそれらはだいたいにおいて同一のものであるという統一性，悪くいえば閉鎖性が重要なのでしょう。

机の上で書類などをマウスを用いて操作するという世界をグラフィックによって実現したマックのデスクトップ環境は，10年前からすでに完成していたと考えることができます。色がついたり，好きなアイコンを容易に並べることもできるようになりましたし，小さい画面用に凝縮されていたのが，最近は広い画面に合わせたデザインに変わってきたことも事実ですが，少なくとも外見的にはそれほど本質的に大きな変化はないように見えます。

しかし，そうはいうものの，実はデスクトップ環境とひと言でいわれているものの実体には，この10年の間にいろいろな変質が起きているのではないでしょうか？

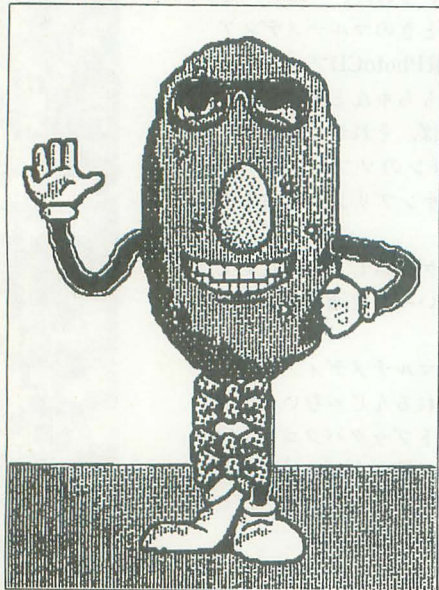
デスクトップ環境の変貌の主役たち

マックの世界には膨大な量の，市販されていない(ここでは「PDS」とひと言でいっていますが)，ソフトウェアが存在します(気に入ったらお金を送る必要のあるものも少なくありませんが)。そして，マックを使いこなしてくると，システムフォルダのなかになんやかやと多くのPDSのファイルを投げ入れることにより，システムを自分の好みに作り変えるようになります。

僕自身も，現在ではそのようなPDSソフトがなければちょっと使えないなと思うほど，システムを自分にとって使いやすいように変更しています。そして，面白いことに，PDSのうちの特にデスクトップ環境に関わるようなソフトウェアには，自分のマックをもっと使いやすくしたいが市販されていないので自力で作るしかないという，並々ならぬ熱意というか意志のようなものが感じられるのです。

マックを買ったときについてくるシステムのデスクトップ環境はこの10年間でたとえそれほど変化がないと仮定したとしても，無数のPDSによって，数多くのマックのデスクトップ環境は時々刻々と変質してきているということは事実です。

そこで，もともとのデザインの出発点である「入力としては主にマウス，出力としてはグラフィックを用いて机の上で書類を操作するというイメージが実現された環境」がどのように変化してきたのかを調べようと思います。

そのためには，僕がこれはいいと思って実際に使っているPDSのソフトウェア群のうちデスクトップ環境に直接影響するもの(全部を同時に使っているわけではあり

ません。衝突してしまうものもあるからです)をみていくのが，いちばん手っ取り早いと思うのです。

ここでは，システムフォルダに入れてシステムそのものを変えるタイプのエクステンション，あるいはコントロールパネル類だけを挙げており，単に外見を変えるようなものやアプリケーションは含まれていません。

Snap-To

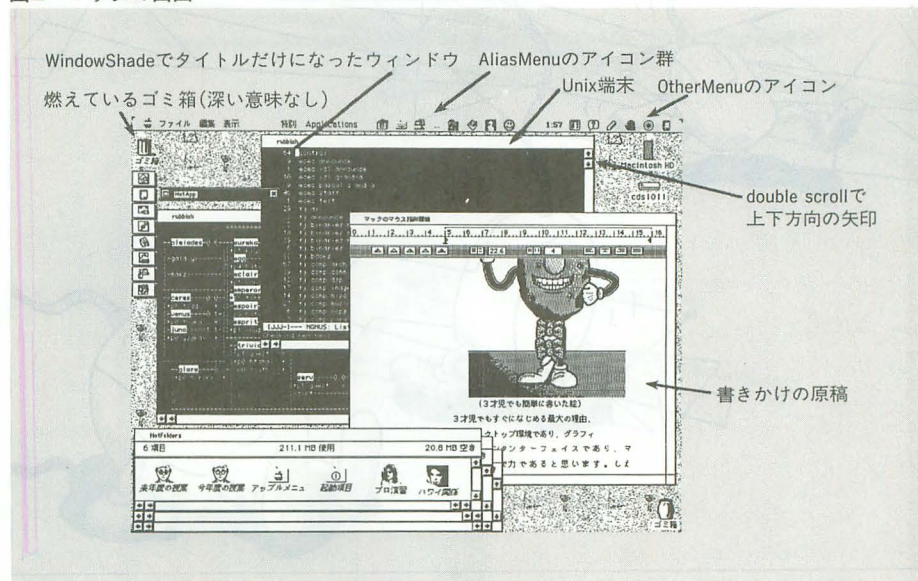
プログラムがメッセージをダイアログボックスと呼ばれる形式で画面に出力し，それに応じてユーザーがマウスによって何らかの答えを返すというのはよく見られる場面です。これをインストールしておくと，そのような場面でボタンを押して選択する際に，最も頻度が高く押されそうなボタン(たとえばOKというボタン)に瞬時に自動的に移動してくれます。

「最も頻度が高く押されそうなボタン」はもともと太枠のボタンになっており，そのボタンに限ってはリターンキーでも代用できるのですが，マウスから手を離すのは効率的でないので重宝しています。

これをインストールしたてのときには，ダイアログボックスが表示されるかされないかの瞬間に，手がきちんと正しい方向に痙攣するようにピクっと動くという事実を知ってしまい驚きました。まったく意識していないのに，現在のマウスの位置からボタンの出るであろう位置への方向に瞬間的に動き出すように，手はマウスによって鍛えられていたのでした。

AliasMenu

メニューバー(画面のいちばん上の帯)のリンゴマークのメニューに好きなファイルを登録して起動できるのはもとからそうですが，これを使うとファインダのメニューバーに新たなメニューを任意個追加できます。「ファイル」とか「編集」とかのメニューの横に好きな名前あるいはアイコンのメニューを加え，それぞれに好きなファイルを登録することができます。

図2 マックの画面

システムフォルダのなかのAliasMenu Itemsというフォルダのなかに番号をふってフォルダを作り，それぞれのなかにファイルを入れると，そのファイル名がメニューに登録されます。よく使うファイルを登録しておけば，フォルダをダブルクリックして目的のファイルを探すという作業はきわめて少なくなります。

OtherMenu

AliasMenuと似ており，メニューバーの右側に新たなメニューを作ります。AliasMenuよりよいところは，階層メニュー(フォルダを登録するとその中身もさらにメニューとして開かれる)をサポートしていること，オープン/セーブダイアログが出ているときに使うとそのフォルダを即座に選択できること，各種の機能をもったプラグインモジュールがついていることなどです。

また，AliasMenuはファインダにいるときしか使えませんでしたが，これはメニューバーに常に表示されるので，ファインダに戻らなくてもすぐに使えます。

MenuDropper

いまのシステムから，ドラッグアンドドロップが可能になりましたが，それをアップルメニュー，OtherMenu，AliasMenuなどに登録されたファイルに対してもできるようにします。たとえば，あるファイルをずるずるっとメニューまでひきずっていき(ドラッグ)，メニューで選択(ドロップ)した項目がフォルダならばドラッグしたファイルをそのフォルダに移動させることになり(オプションキーを押しながらならコピーし，コマンドキーを押しながらならエイリアスを作ることになる)，メニューで選択した項目がプログラムならドラッグしたファイルをそのプログラムでオープンすることになります。

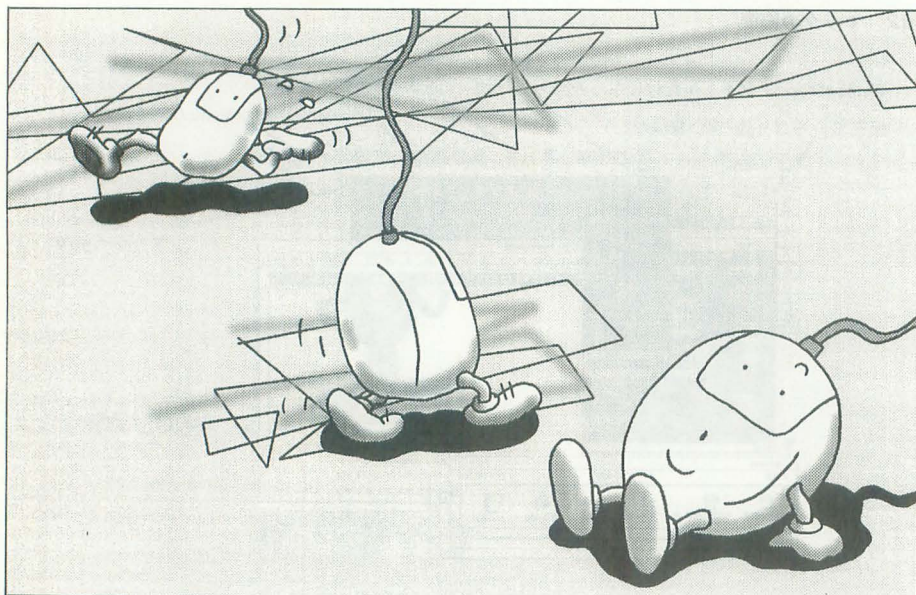
これにより，プログラムを実行するのがワンタッチでできるようになっただけでなく，ファイルの移動なども即座にできるようになりました。

TearOFFs

メニューをメニューバーから取り外して好きなところにおくことができます。よく使うメニューを切り離して近くにおいておくと便利です。

Click, there it is!

オープン/セーブダイアログが出ているときに背景にあるファインダのフォルダウィンドウ(ディレクトリ)をクリックすると，そのディレクトリが選ばれます。ファイル

そしてマウスは運動不足になる

illustration：Haruhisa Yamada

をセーブするための場所をダイアログボックスのなかでたどっていく手間を省くことができます。

Default Folder

アプリケーションごとにファイルを読み書きするデフォルトのフォルダを指定できます。

WindowShade

タイトルバー(ウィンドウ上部の帯)をダブルクリックすると，そのタイトルバーだけを残してウィンドウが消えます。もう一度ダブルクリックすると元に戻ります。たくさんのウィンドウを開いておくときに，とても重宝します。

ZoomBar

Xウィンドウにおけるアイコン化と似ており，Zoomボタンをクリックするとそのウィンドウを表す小さなタイトルバーが指定の位置に整列されます。よく開くウィンドウを登録しておくと，普段は場所をとらないので便利です。

Hierarchy

ファインダのフォルダウィンドウ(ディレクトリ)で，タイトル部分をクリックすると，オープン/セーブダイアログボックスで上端に設けられる親フォルダ選択メニューと同じものが現れます。親のディレクトリに移動したいことは案外あるものですが，それが簡単にできます。

DragAnyWindow

普通は，前に出ているアクティブウィンドウしか移動できません。これをインストールすれば，特定のキーを押しながらどのウィンドウでもドラッグして移動することができます。

DoubleScroll

スクロールバー(ウィンドウの右側の帯)の矢印ボタン(押すと上あるいは下にスクロールする)は上向きのは上に1個，下向きのは下に1個しかありませんが，これにより上下ともそれぞれに上下方向の矢印が現れます。したがって，上へのスクロールと下へのスクロールをすばやく切り換えることができます。

マウスの運動量を減らすために

小気味よいこれらのソフトウェア群を見てくると，それらの目的はただひとつであることがわかります。それは，「マウスの運動量を少しでも減らす」ということです。

確かにマックに向かっている時間のうち，文字を入力している時間を除けば，多くの時間は手首を動かしてマウスを走らせている時間です。その走行距離を減らすことがダイレクトにマック操作の効率化につながるというわけです。

紹介してきたPDSによって実現される拡張機能は，当初考えられてきたデスクトップという概念とはなんら関係のない，というかむしろ想定されてきた操作法を意味のないものにするという性格の強いものです。それは，初心者にもなじみやすいように，画面を机の上のように見せかけるといったものとはむしろ逆行する，「効率優先のマウス指向主義のあくなき追求」とでもいったところでしょう。

当初考えられていたデスクトップ環境のもつ性質のうちのどこが効率優先マウス指向主義と対立する問題点となってきたのでしょうか？

1)　マウスの動きも現実世界と同じように連続的であり，パッと目的地につくことができない。

2)　フォルダの階層構造をたどって目的のファイルにたどりつくのに手間がかかる。

3)　大量のファイルやディレクトリの存在によって画面が一杯になった場合の効率的操作法が用意されていない。

紹介したPDS群を眺めると，少なくともこの3つの問題点が浮き彫りにされます。もちろん，初心者の人にとってはこのようなことはさして深刻な問題ではないでしょう。しかし，使い込んでくると仕事のはかどり具合に大きく影響してきます。まどろっこしくなってくるのです。ところが，これらのPDSを入手しシステムフォルダに置くだけで，画期的に環境が改善するというわけです。要するに，デスクトップ環境を表面上には残しつつ，実質的には効率優先のマウス指向環境に変貌するのです。

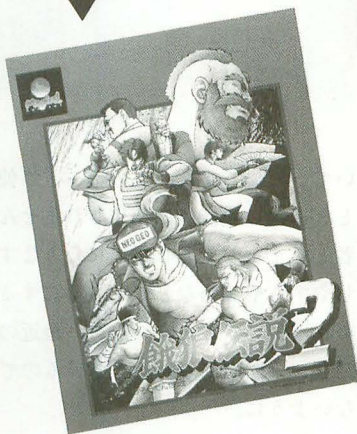
でもどうなんでしょうか？　指しゃぶりの3歳児でもすぐになじんで使いこなし，同時にプロでも快適に仕事ができるという両面性をもつマックは，実は案外ぎりぎりのところにいるような気がしませんか？

愛読者プレゼント

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1994年3月18日の到着分までとします。当選者の発表は1994年5月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号のほかの懸賞には当選できない場合がありますので，ご了承ください。

1

魔法株式会社　☎078(261)2790

餓狼伝説2

X68000用　5"2HD版

9,800円(税別)

2名

1～2月号に連続で紹介したNEO・GEOからの移植アクションゲーム。初回限定のオマケの4つボタンパッドが使える次回作の発売も決定しました。

2

カプコン　☎03(3340)0750

ストリートファイターII ダッシュ

X68000用　5"2HD版

3名

12,800円(税別)

人気絶頂につき，今月号からキャラ別に紹介していきます。コネクタ同梱なので2つめが欲しい人も応募してね。

3

ソフトバンク　☎03(5642)8101

GCCによるX680x0ゲームプログラミング

3,600円(税込)

5名

1月号のペンギン情報コーナーで紹介したX68000活用本。「C Magazine」の連載に加筆・修正されたものです。2枚の付録ディスクにはGCCの実行環境などが収録されています。

4

上昇気流 vol.5　5名

毎年恒例の高橋哲史氏作「上昇気流」。早いものでもう5冊目です。入手していないファンには重大なお知らせが……詳しくは編集後記を読んでね。

5

ファミリーソフト　☎03(3924)5435

ファミリーソフト特製

卓上カレンダー

非売品

5名

「マッドストーカー」のファミリーソフト製の今年のカレンダー。アニメファンには嬉しい絵柄ですね。3.5インチフロッピーディスクサイズ。

1月号プレゼント当選者

1スーパーリアル麻雀PII＆PIII　(新潟県)金子卓司　(長野県)伝田佳史　(奈良県)開口嘉雄　2ぬりぐすり　(宮城県)佐藤友一郎　(新潟県)渋谷洋明　(千葉県)杉本英也　(愛知県)増川一詞　(福岡県)梶原　修　3中国茶　(神奈川県)須川雅志　(石川県)佐渡詩郎　(岡山県)須田周作　4ザウルス出現！　(栃木県)長崎　洋　(埼玉県)横山紘一　(静岡県)大口英臣　(愛知県)佐藤　真　(大阪府)竹内孝雄　5MIYA-NET特製カレンダー　(福島県)鈴木俊雄　(千葉県)村岡　篤　(埼玉県)横堀正敏　(岐阜県)田川和義　(京都府)西尾征訓　6ソフトバンク卓上カレンダー　(埼玉県)加藤昌宏　(神奈川県)柴野美由生　(石川県)新井由之　(香川県)西池陽一　(愛媛県)中矢史朗ほか25名　(敬称略)以上の方々が当選しました。ソフトバンク卓上カレンダーは20名の予定でしたが，ご好評につき30名にプレゼントいたします。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

SIMMを使った増設メモリ

拡張メモリボードXsimm10

Kioi Makoto 紀尾井 誠

Xsimm10はSIMMを使用したメモリボードです。

SIMM(Single Inline Memory Module)というのは特定の形状のメモリボードの種類を表す言葉です。要はカタチの問題ですから写真を見ればだいたい感じはわかってもらえると思います。

で，SIMMのなにがいいかというと，それは「安い」のひと言に尽きるでしょう。基板上にはメモリしか載っていないのでそもそも安く，メジャー機種用に大量生産されているのでさらに安いという図式が展開されています。だいたい1Mバイトのもので5,000円くらい，4Mバイトで16,000円くらい，安いところを探せばそれなりに……といった感じです。

すでに同様にSIMMを使ったX68000用のメモリボード（？）としてPOLYPHONがありますね。

Xsimm10ではSIMMを4枚まで差し込んで使うことができます。この手のSIMMは2個単位で使うことになりますので，

1MバイトSIMM 2枚
4MバイトSIMM 2枚

という構成にすれば10Mバイト分の増設メモリとなり，本体とあわせてめでたく12Mバイトフル実装という状態が完成します。同様に，使用するSIMMの組み合わせによって2M，4M，8MバイトのRAMボードに構築することもできます。

とりあえず1枚のボードで10Mバイト分増設できるというのは魅力でしょう。しかも，SIMMの買い足しや買い換えができるので気軽にステップアップさせることができます。昔は12Mバイトフル実装をするためにRAMボード3枚（あるいは4枚）と拡張I/Oボックスが必要だったことを考えれば(最高478,800円！)，ずいぶん身近になったものです。

Xsimm10を使う

Xsimm10はメモリボードですので単体での使い方……といったものはありません。

Xsimm10　18,000円（税別）
東京システムリサーチ　☎0425(28)1824

拡張スロットに取り付け，SWITCHコマンドでメモリの設定を変えれば増設しただけ快適な環境が得られます。

問題になるのは，このボードはメモリなしの状態で販売されますので，SIMMは自分でみつけてこなければならないということです。

使用できるSIMMはMacintosh用またはIBM PC互換機用の30ピンのものです。PC向けのものにはパリティチェックのためチップが1個余分についているのですが，このボードで使用するときにはこの機能は使用されませんので両者ともまったく同じ扱いになります。入手のしやすさや価格などでどちらにするかを決めればいいでしょう。なお，最近の機種は72ピンのSIMMを採用しているので間違えないように。

現在売られているものだとアクセスタイム80nsというのが一般的です。60nsくらいのものまで入手できますが，普通の人が使うならだいたい100ns以下のSIMMであればほとんど性能は変わらないはずです。

10MHz機での使用

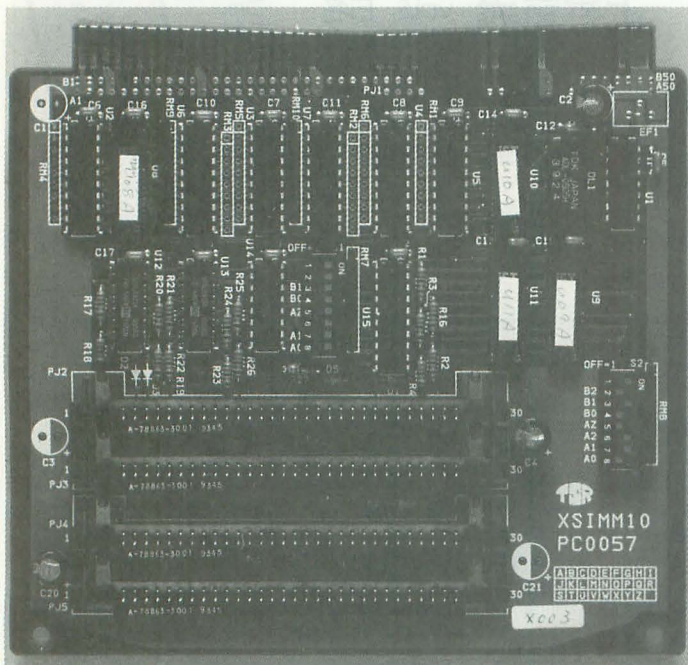
10MHz機で使う限り性能的にはまったく問題はないでしょう。スロットもメモリでひとつはつぶれるのが当たり前ですから，1枚でフル実装できるこのボードは理想的といえます。

あとは値段の問題だけ。ボードが18,000円，2Mバイト載せると3万円弱，4Mバイト仕様で4万円弱，8Mバイト仕様で5万円弱，10Mバイト仕様なら6万円弱……。いずれもいかに安いSIMMを入手するかというのが問題です。

秋葉原や日本橋に近ければ足で探すのが最良でしょう。地方の人はMacintosh雑誌や「トランジスタ技術」などの広告を参考に通販を利用するのがよいでしょう。なお，こういったものはブランドや販売店によってかなり価格に幅がありますし，変動相場性のものですので記事中の価格もあくまで参考程度にとどめておいてください。

高速機への対応

さて，このメモリボードのウリのひとつに高速機への対応があります。これは普通の人にはあまり関係ないことなのですが，X68000をクロックアップ改造している場合にもノーウェイトで対応できるというこ

30ピンSIMMの例（Macintosh用）

とを意味します。

こういった改造を行うと拡張スロット内のメモリについては，最悪の場合動作しないか，動作してもかなりのウェイトが入ってしまいます。内蔵RAMでは高速に動作しても，拡張RAM部分ではぐっと遅くなり，往々にして改造前のほうが若干速くなることもあります。

このボードでは高速なSIMMを使い（70ns以下)，最適なタイミングでアクセスすることにより，18MHzまでの改造機でもノーウェイトで動作することを保証しています。

ただ，じゃあどんな改造でも大丈夫かというと，そういうわけでもなく，拡張スロットにちゃんとバスクロックが供給されている場合のみです。当たり前ですけど。

もう少し具体的に書くと，「旧型の10MHz機をなにも考えずにクロックアップしている場合」か「X68000XVIのスロットにクロックアップしたものを送っている場合」がこれに当たります。「10MHz機をちゃんと安全（？）に改造している場合」や，「X68000XVIをなにも考えずに改造している場合」はこれに当てはまりません。

さらにいえば，10MHz機を安全にクロックアップしてある仕様のX68000XVIやX68000XVIをなにも考えずに改造してあるREDZONEでは，拡張スロットにクロックアップしたバスクロックが供給されていませんので，ノーウェイト動作は実現できません。また，スロットへ高速クロックが供給されている場合，その他のボード類が動作しない可能性が高くなります。

だいたいおわかりでしょうか？

編集部にはこういった改造機がありませんのでテストはしていませんが，RAMが追いつけば特に問題はないはずです。これまでの市販RAMボードは拡張スロット（必ず10MHzで動作）を前提に作成されていたため，改造機ではRAMが追いつかなかっただけの話ですから。

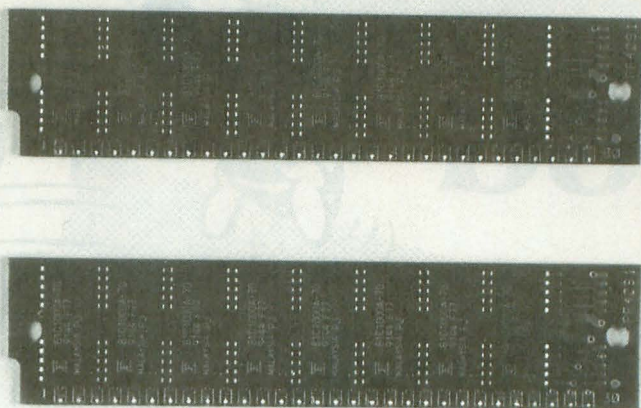
すでにクロックアップ改造を行っている人には朗報かもしれません。しかし，こういったものが使えるからといって安易にクロックアップ改造に走るのは危険です。X68000XVIの24MHz化程度ならまだしも，10MHz機ではかなりうまくやらないとシステムクロックごとクロックアップしてしまうことになります。こうなるとハードウェアの負担が大きくなりますので，動作していても突然動かなくなるということも結構あるようです。壊す覚悟でやっている人は無理には止めませんが……。

X68000XVIでは？

では，X68000XVIでこのボードを使用した場合はどのようになるのでしょうか？

X68000XVIでは本体のCPUは16.6MHzで動作していますが，スロット上のRAMは10MHzで動作することになりますので，かなりのウェイトが入ってしまいます。仮にプログラムのすべてが拡張RAM上で動作するとすると，単に10MHzで動作させたときより14％遅くなります（70nsRAM使用時）。

ということで，個人的にはX68000XVIユーザーやREDZONEユーザーならスロット節約の意味も込めて，まず内蔵RAMで増設することをおすすめします。確かに「10MHz時より遅いRAM」とはいっても実使用上で気になることはほとんどありませんし，いくら遅かろうが，メモリがあるのとないのでは天と地ほどの環境差があるのですが，少なくとも内蔵RAMだけで4Mバイト以上にはしておくべきでしょう。

まあ，メモリ10Mバイト以上に増設するならどのみちスロット用の増設RAMが必要ですから，うまくすれば割安になる可能性もあります。30ピンSIMMならばトラ技の広告あたりで見ても1MバイトSIMM2枚で8,000円からありますので。

また，Macintoshの場合，メモリを拡張するとたいてい古いSIMMが余るようになっていますので，しばらく前からMacintoshユーザーをやってるような知り合いがいたら声をかけてみるのも手です。

12Mバイト時代はくるか

一般的にX68000は通常使用では6Mバイトあれば必要十分な環境が揃います。では，どのようなときに12Mバイトのメモリが有効になるかというと，それは大規模なプログラムを作ったり（特にグラフィック関係），CGAに手を出したり，とりわけSX-WINDOW環境での作業時ということになるでしょう。MATIER-EXを使うときもメモリは湯水だと思ったほうがいいですね。

テンポラリファイルなどを生成したりするRAMディスクは大きめに取りたいところですし，ディスクキャッシュもつけたほうがよいでしょうし，SX-WINDOW環境ではフォントキャッシュも贅沢にいきたいものです。

特にSX-WINDOWでは開発ツールやEG Wordといった大モノが予定されていますのでオーバー6M環境が切実に必要になってくることも予想されます。この製品もそういった時代の要請によって生まれたメモリなのかもしれません。12Mバイトが当たり前になる日はくるのでしょうか。

Another
CG
World

今月
さむいよ〜
おふとんから
出たくないよ〜

今月の目標
いつもニコニコ
笑顔
で仕事
ん？

今月
いつ
笑顔
で仕事
んーいいにおい
もう春なのかねー

よーし ここはひとつ
花でも育てて
みるか！

まいにちまいにち

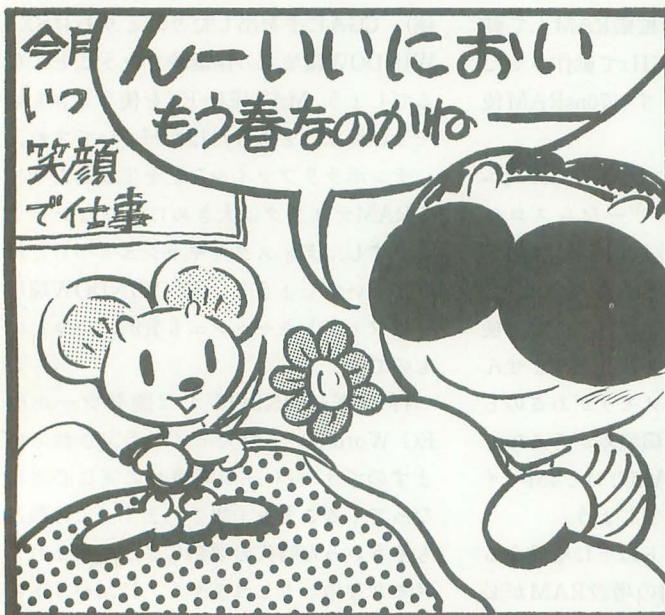
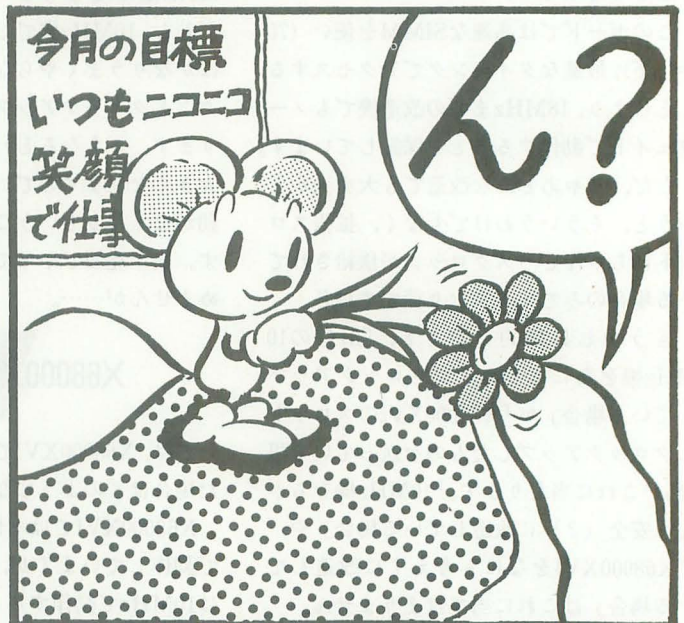
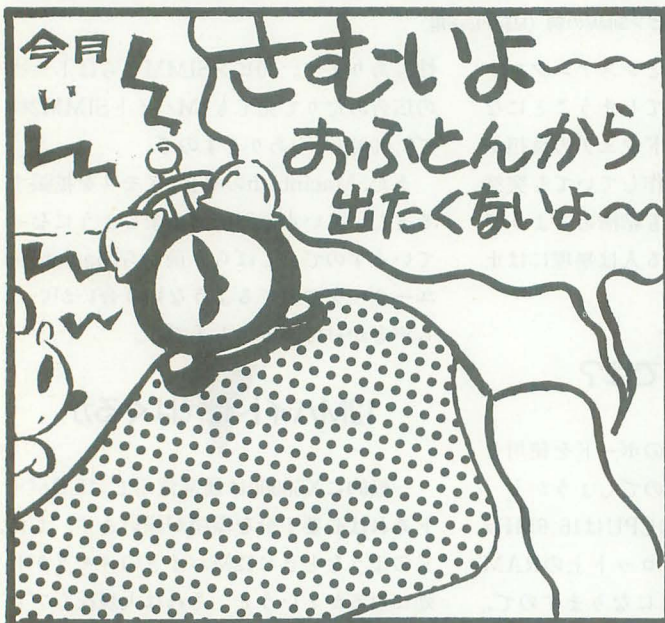
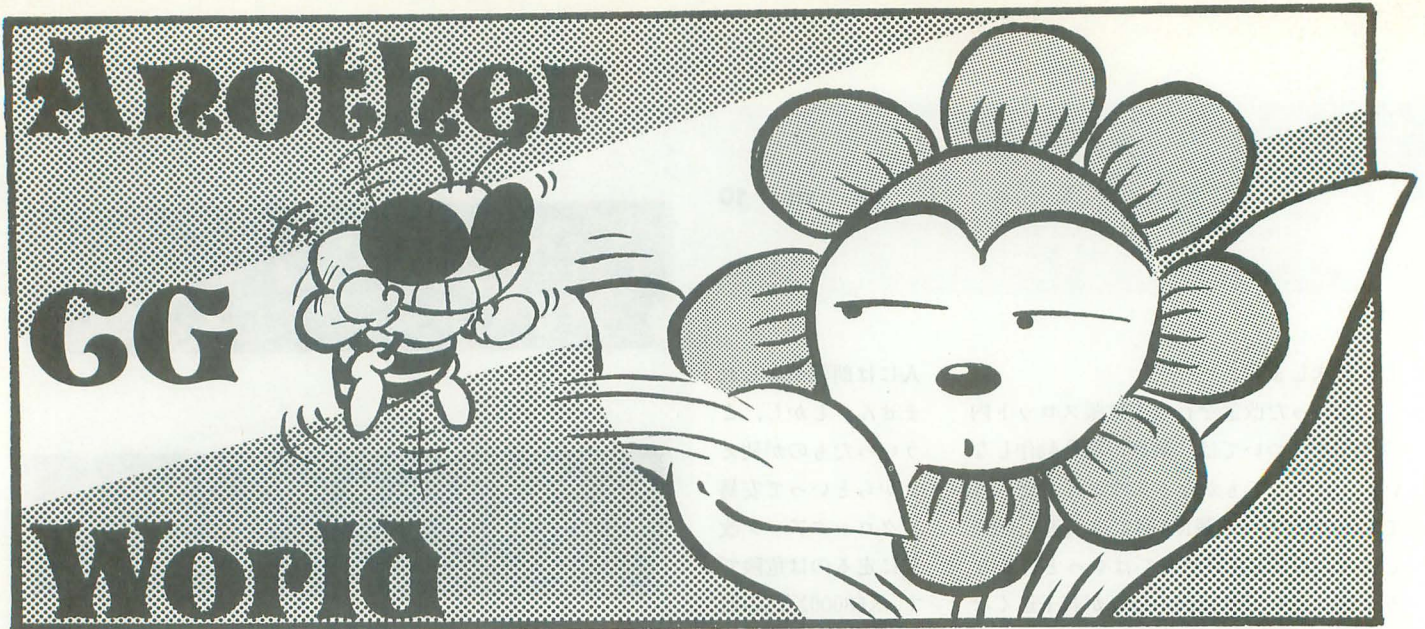
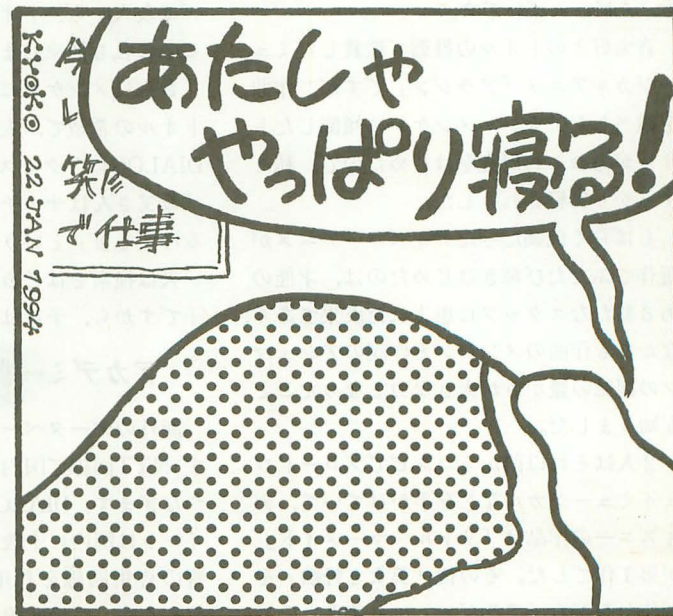
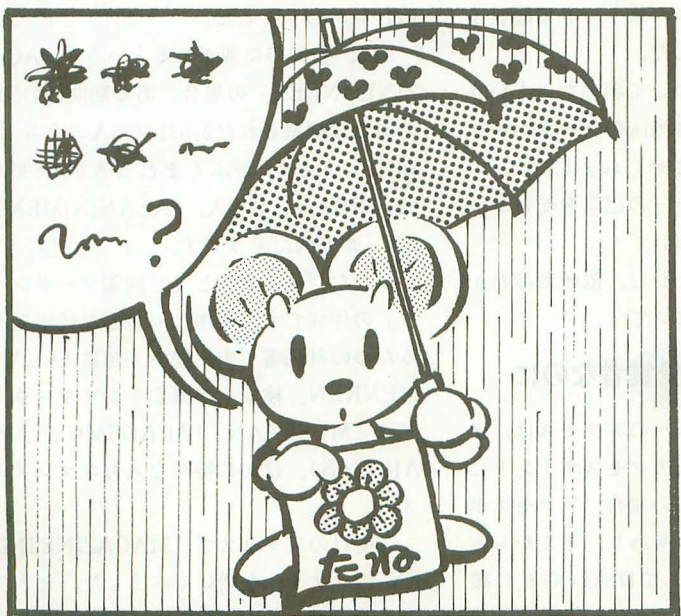
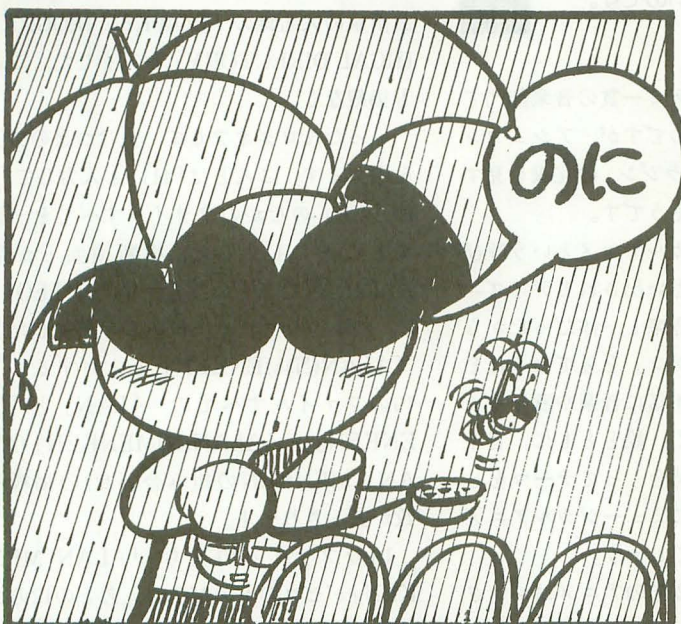
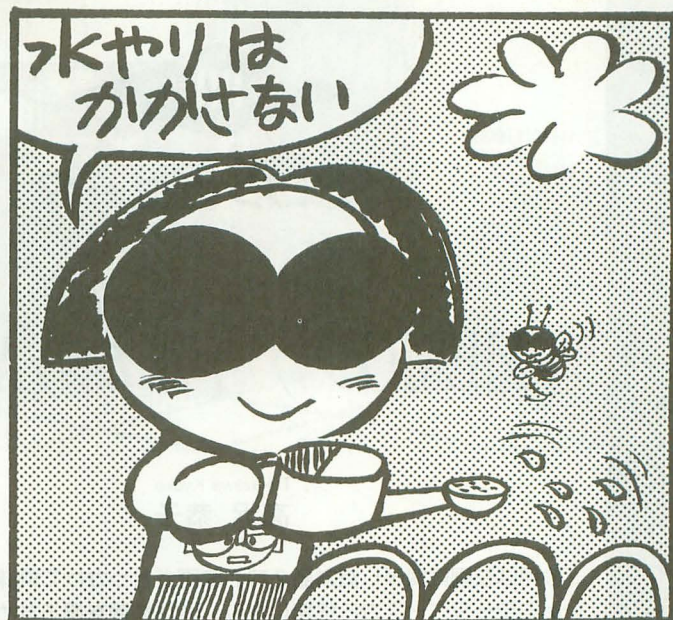
水やりは
かかさない

のに

なんでここだけ
花が咲かない
のよ——
NOTHING

?
たね

KYoKo© 22 JAN 1994
あたしゃ
やっぱり寝る!

第89回

猫とコンピュータ

MEMOファイルからA&B

Takazawa Kyoko
高沢 恭子

今月は映画に関係したお話2つ。映画がきっかけで興味をもったことをデータベースから検索したり，逆に情報を得るために映画の中を探してみたキョウコさん一家。興味の広がりは尽きないのです。

毎回の連載に載せきれずにこぼれてしまった話が，メモになってたまっています。だんだん鮮度が落ちていき，いずれ捨てられていくのですが，そのなかからいくつかを拾ってみようと思います。たくさんあるので，アルファベット順に頭文字と関わりのある話を選ぶことにしました。残りはつぎの機会につづけます。

A アラン・メンケン

作曲家アラン・メンケンの名前を知ったのはディズニーのアニメーション「アラジン」で，メンケンの強力なパートナーだったのが，いまは亡き作詞家ハワード・アシュマン。そして2人で獲得したアカデミー賞。Aがいっぱいです。

音楽好きのトオルの推薦で鑑賞したミュージカルアニメ「アラジン」ですが，作曲を担当したアラン・メンケンに傾倒したトオルがあれこれ研究をはじめたので，私もすっかり影響されました。

しばらく低調だったディズニーアニメが近作でふたたび輝きはじめたのは，才能のある新たなスタッフに恵まれた成果であり，なかでも作曲のメンケンと作詞のアシュマンの創意の豊かさが大きな力となったことも知りました。

2人はそれ以前からコンビでブロードウェイミュージカルなどを手がけていて，ディズニーの作品は「リトル・マーメイド」が第1作でした。その後「美女と野獣」を制作しながら「アラジン」にも着手，そしてこの3作ともアカデミー賞の音楽部門で数々の賞を獲得したのですが，アシュマンは「美女と野獣」「アラジン」の完成を見ずに急死してしまったそうです。

はじめはミュージカルアニメという呼び名がフシギでした。思い出されるディズニーの作品は，どれも音楽と一体だったからです。しかし「アラジン」を見てなるほどと思いました。おとぎ話に音楽や歌をそえるようなものではなく，絵とともに，音楽が一貫した表現の姿勢をもってテーマをしめそうとするところにミュージカルアニメと呼ぶものがあるのでしょう。

「アラジン」以前の2作のアニメはレンタルビデオ店から，音楽を収録したCDやテープは友人から，トオルが借りてきて鑑賞するのに私も便乗しました。

さらにメンケンについて調べたいというトオルの希望で，夫は米国のデータベースDIALOGにアクセスさせられました。

「お父さんはサーチャーの認定を受けているんでしょ」というわけです。

夫は検索をはじめました。依頼者はわが子ですから，予算はゼロです。

アカデミー賞受賞者なのに

海外のデータベースへのアクセスは，まずNTT回線で国内のアクセスポイントにつなぎます。DIALOGの場合，アクセスポイントは国内に十数カ所あり，そこから丸善の専用回線を利用してDIALOG社に接続します。この回線がパケット通信で，アクセスしているあいだ1分あたり90円ほどの料金がかかります。DIALOG内での検索はファイルの使用時間と出力件数で料金が算出され，ファイルごとに料金がことなります。1度のアクセスでかかる費用は，NTT，丸善の各回線使用料と，DIALOG社の情報料を合算したものです。

人物の情報ファイルは「PEOPLE」のジャンルにあつめられています。そのなかから，これと思うものに当たりをつけてファイルを開いてみるのです。まず，アメリカの人名録では一番有名な「BIOGRAPHY MASTER INDEX」を見ました。「ALAN MENKEN」も「MENKEN ALAN」もなぜかゼロです。つぎに科学者，政治家，芸術家の人名録の「BOWKER BIOGRAPHICAL DIRECTORY」を見ました。ここも掲載なし。

いったんログオフして，夫はつぎの方法を考えます。こんどは「411」と入力して「PEOPLE」のなかのファイルすべてを一覧することにしました。横断検索とかDIALINDEXなどという方法です。「PEOPLE」には19のファイルがあると出ました。

「ALAN(N)MENKEN」と入力すると，19のファイルすべての「ALAN」と「MENKEN」と「ALAN(N)MENKEN」(ALANとMENKENの組み合わせ)の掲載件数が表示されます。

19のファイルのうち，「ALAN MENKEN」がゼロのものが7つ，あとの12のファイルには最多で46人，最少で1人まで掲載件数がありました。

37人(件)の掲載があるという「MAGAZINE INDEX」の場合，ある期間内の500の雑誌に掲載された2,531,455人のうち，名前に「ALAN」がふくまれる人が8,690人，「MENKEN」が39人，「ALAN(N)MENKEN」が37人だそうです。

ふたたびログオフして，「検索ワークシート」の作成です。効率よく必要な情報を得るための計画書です。検索主題は「ALAN MENKEN」。検索上の概念を示すキーワードは「MUSICIAN」「ACADEMY」「AWARD」(賞)。ほかに特殊な入力コマンドのメモ。

3度目のアクセスで，「MAGAZINE INDEX」のファイルを開いてみました。全リストのなかで「ACADEMY」で検索すると，

該当者が4,822人。「AWARD」では4,712人。両方に関わる人は198人。そのなかの「ALAN(N)MENKEN」は，なんとゼロ。「ACADEM???」の入力でアカデミーに関する人に条件を広げると8,198人，そのなかにも該当者はゼロでした。

費用を考えずにすべてのファイルを閲覧すれば，なんらかの情報が得られるにちがいないのですが，いったん海外へのアクセスは打ち切りました。

アシスト社の「WHO」

つぎに国内のデータベースとして朝日新聞のデータベースを調べたのですが，あったのはメンケンとアシュマンのミュージカルを日本の俳優たちが上演するという2年前の記事だけでした。

最後の手段として，NIFTY-Serveの「サーチャー倶楽部」にアクセスして，メンバーのかたたちに助けを求めることにしました。DIALOG社のいくつかのファイルに残念ながらヒット(的中)がなかったことや，ダウンロードの記録も掲載しました。サーチャーはみずからクライアントになるのも仕事のようです。

その日のうちに女性サーチャーのKさんから情報をいただきました。日外アシスト社の「WHO」(人物・人材情報)のファイルにありますよとのこと。

さっそく日外アシストにアクセスして，「WHO」のファイルの検索です。検索語の一覧表示「L」で「メンケン，アラン」を入力すると1件あり，やっとアラン・メンケンのくわしい情報を1つ入手できました。

こうしてみると，メンケンの本国アメリカでの膨大な数のデータから，作曲家でアカデミー賞を得たメンケンをさがしだすより，日本に1人しか紹介されていないメンケン氏をはじめから拾うほうが近道と思われます。日外アシストにたった1人いたアラン・メンケンなら「アラジン」の彼にまちがいないでしょうから。

ただトオルの希望は，本国での評価や英文での表現を見たかったのだそうで，すこし残念でした。46人(件)も記録のあったファイルの46人ぶんの検索結果にすべて目をとおしたなら，まちがいなくあのメンケン氏の記事がたくさんあったことでしょう。あるいはかえって「1人」だけ掲載があった「NEWSEARCH」というファイルを開いてみればよかったのかもしれませんが，海を越えての情報獲得は時間との勝負ですし，お父さんの支払いではトオルもぜいたくはいえません。

それでも日外アシストの情報で，彼がニューヨーク大学医学部の卒業であることや生年月日，出身地もわかりました。そのほかの情報は，やはりアシュマンとのコンビでの作品や，ディズニー作品でのアカデミー賞に関することが主でした。

夫の検索で私も海外のデータベースをはじめて見学しました。こんな大きなデータベースがいくつもあるとしたら，サーチャーたるもの毎日トレーニングにはげまなくては，顧客の注文と予算に応じた，たしかなデータを得ることはむずかしいでしょう。

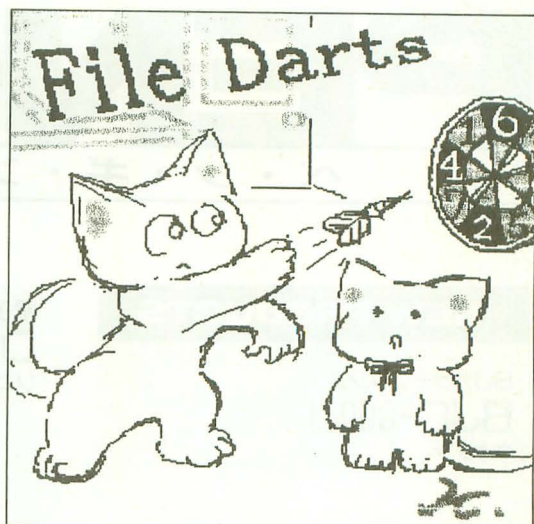
B「Back to the Future」

ジョージ・ルーカスとともに，「映像の魔術師」と呼ばれるスティーブン・スピルバーグ。人気を分けあう2人には話題作も多く，テレビで放映されるたびに何度でも見てしまいます。

なかでもその奇抜さ，痛快さのとりこになったのが，スピルバーグの「バック・トゥ・ザ・フューチャー」でした。第1作が公開された1985年，小学生だったトオルは劇場ですでに見ているのですが，私はここ数年テレビで放映されるようになってからはじめて見たのです。

お話づくりの面白いこと，登場人物の魅力的なこと，SFなのに現実感が豊かなことなどにひかれて，どうしても原語版ビデオがほしくなり，まず「Part I」を買いもとめました。やはり俳優たちのほんとうの声を聞かなくては，作品を知ったことにはなりませんから。

それからしばらくして，たいへんショッキングで悲しいことですが，アメリカに留学中の日本の高校生が，相手の誤認から銃で撃たれて死亡してしまうという事件が起こりました。そして「Freeze」という言葉が，日本じゅうに広まりました。

新聞記事に，アメリカ映画では会話のなかによく聞かれる言葉であり，「バック・トゥ・ザ・フューチャー」などはその例であると書かれていました。「Freezeがほんとうにあったかしら」と，またビデオをくりかえし見ました。

illustration：Kyoko Takazawa

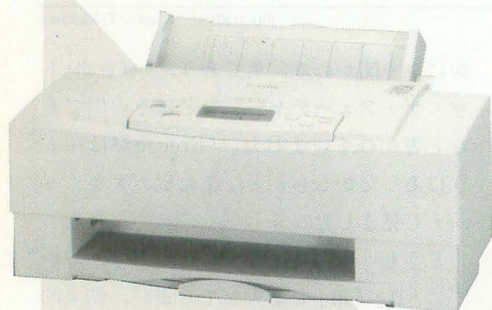
どうも「I」にはないらしいので，ついでに「II」「III」も購入してしまい，結果的には全巻がそろいました。ただし，じっさいに「Freeze」が聞かれたのは，「II」のなかでただ1度だけだったと思われます。

アメリカ映画を原語で鑑賞すると生きた英会話の勉強になるというのは，いままでもよくいわれていたことですが，銃を向けられたときの習慣まで学べるというのでしょうか。私の耳には「バック・トゥ・ザ・フューチャー」のなかで1度しか「Freeze」が認められなかったことで，すこしホッとしているのですが。

もうひとつ，1991年のスピルバーグの作品「フック」で，成人してしまったピーターパンを演じたロビン・ウィリアムズは，ディズニーの「アラジン」で「魔神ジーニー」の声と歌を演じた人です。この熱演でつくりあげた魔神のキャラクターのすばらしさに，彼は第50回ゴールデングローブ賞の特別賞を受けたそうです。

それにしても「バック・トゥ・ザ・フューチャー」は，何回見ても飽きることがありません。愛すべきブラウン博士の大発明タイムマシン「デロリアン」で時計台を拠点に時空を行き来する愉快さ。話の構成とツジツマ合わせの完璧さに，この3編のシリーズはどれからはじめに構想されたのだろうと考えてしまいます。

NEW PRODUCTS

BJカラープリンタ
BJC-600J
キヤノン

BJC-600J

キヤノンはBJカラープリンタ「BJC-600J」を発売した。

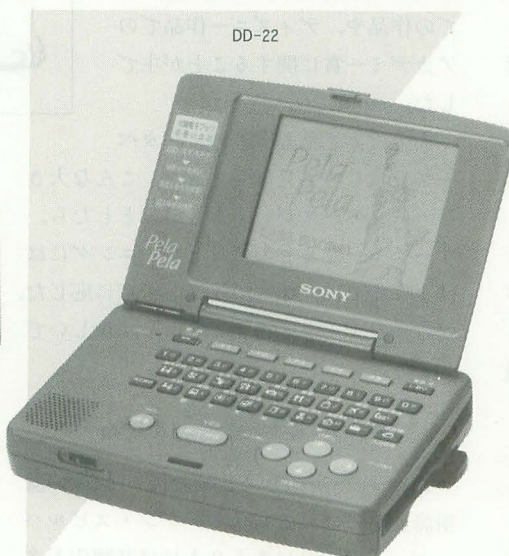
本機は印字方式にシリアルバブルジェット方式を採用し，解像度は360dpiとなっている。従来のインクより速乾性のものを開発したことで，コピー用紙などBJ専用紙以外の紙での印刷が可能になった。カラー印刷に使用する4色のインクカートリッジはそれぞれ独立しているので，消耗したインクカートリッジのみの交換ですむ。カラー印刷の場合は，A4原稿（各色7.5％）を約18.1円，モノクロ印刷の場合はA4原稿（ANK1500文字）を約2.3円で印字が可能。

印字速度はANKで最高240cpsを実現した。また，印字モードは速度優先のHSモード，仕上がり優先のHQモードが選べる。フォントは標準で，明朝・ゴシック・毛筆の3書体を内蔵している。標準装備のオートシートフィーダを利用することで，A4の場合で約100枚，官製ハガキで約40枚が給紙できる。

大きさは，410mm（幅）×253mm（奥行）×184mm（高さ）となっている。

価格は，120,000円（税別）。

〈問い合わせ先〉
キヤノン販売㈱　☎03(3455)9544

電子ブックプレーヤー
DD-22
ソニー

DD-22

ソニーは電子ブックプレーヤー“PelaPela”「DD-22」を発売した。

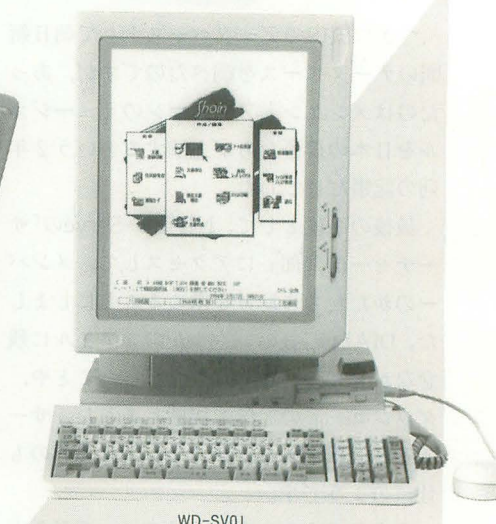
本機には，専用電子ブック「JTBの海外旅行英会話」が同梱されている。同ブックは海外旅行，海外出張で使用頻度の高い英会話表現3,037文例を収録している。その全例文はネイティブスピーカーの生の発音で収録され，内蔵のスピーカーで音声を再生させることが可能。「買い物」「ビジネス」「観光」など17場面の文例をダイレクトキー方式で素早く選ぶことができる。

専用電子ブック以外のさまざまな電子ブックも再生可能。画面は16文字×10行の4インチ（256×200ドット）液晶ディスプレイを採用している。出力端子はヘッドフォンとビデオの2種類がある。電源はアルカリ単3電池4本で約4時間再生できる。大きさは，166mm（幅）×113mm（奥行）×40mm（高さ）で重さは445g（電池なし）。

また，別売りの「DD-CD1」（800円，税別）を使うことにより，音楽用シングルCDの再生が可能になる。

価格は，39,800円（税別）。

〈問い合わせ先〉
ソニー㈱　☎03(5448)3311，06(251)5111

オフィスワープロ書院
WD-SV01
シャープ

WD-SV01

シャープはオフィスワープロ書院「WD-SV01」を発売した。

本機は，業界で初めて文書を画面に表示した印刷レイアウト図を見ながら検索できるビジュアルサーチャーを搭載した。また，「PI-3000」「WS-250」「PV-F1」（いずれもシャープ製）で作成したデータを光通信で取り込むだけでなく，プリンタへの送信印刷が可能になった。表現豊かな文書作成を助ける「書院スーパーレイアウト」「アート倶楽部ビジネス版」などが搭載されている。他社文書呼出機能も従来の3社から5社に増えた。

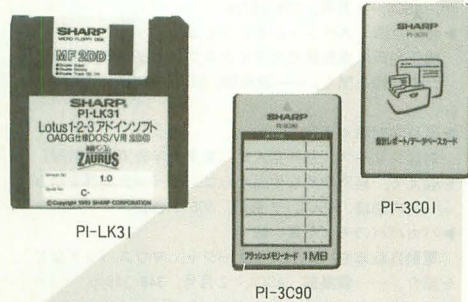
画面は960×1,280ドットの高解像度液晶を採用し，A4縦サイズをフルページで表示できる。辞書は約1,000,000語（約462,000語のAI用例を含む）を搭載。170Mバイトのハードディスクドライブを内蔵している。

別売りのスキャナ「JX-325M」を接続して，定型フォーマットを読み込むと記入欄を自動的に認識し，文書を作成するので，定型文書の印刷が簡単に行える。

価格は，1,200,000円（税別）。

〈問い合わせ先〉
シャープ㈱　☎06(621)1221，043(299)8210

PI-3000用周辺機器
PI-3C90/3C01/LK31
シャープ

PI-LK31　PI-3C90　PI-3C01

シャープはPI-3000（ZAURUS）用の周辺機器「PI-3C90」「PI-3C01」「PI-LK31」の3種類を発売した。

「PI-3C90」は，フラッシュメモリカードでPI-3000のメモリを増やすためのもので，メモリ保持の電源は必要ない。メモリ容量は1Mバイト（ユーザーエリア約780Kバイト）でスケジュールデータなら約19年分，名刺データなら約4,000人分を記憶できる。

「PI-3C01」は，集計レポート/データベースカード。集計レポートは，価格表や実績表などのさまざまな集計表を組み込んだレポートの作成を可能にした。データベースはカード型で，顧客管理，販売実績など8種類の定型フォームが用意されている。オリジナルのフォーム作成，複数条件（3つ）の検索や検索結果に対しての検索（絞り込み検索）もできる。レポートやデータベースの内容は別売りのケーブルを使いパソコン用プリンタなどで印刷できる。

「PI-LK31」は，Lotus 1-2-3アドインソフトで，「PI-3C01」とパソコンソフトのLotus 1-2-3とのデータ交換が可能になる。対応パソコンは，PC-9801シリーズ（VX以降），OADG仕様のDOS/Vパソコンである。通信には別売りのケーブルが必要で，光通信によるデータ交換も可能。

価格は，「PI-3C90」が28,000円，「PI-3C01」が25,000円，「PI-LK31」が8,000円となっている（それぞれ税別）。

〈問い合わせ先〉
シャープ㈱　☎06(621)1221,043(299)8210

ナイスパートナー手帳
PA-EZ2
シャープ

PA-EZ2

シャープはナイスパートナー手帳「PA-EZ2」（愛称：Page）を発売した。

本機は，閉じたときに117mm（幅）×81mm（奥行）×10.9mm（高さ）で重さが約120g（電池含む）と，ポケットに入るサイズの電子手帳である。主な特徴としては，手帳に自分だけのキャラクターを設定（似顔絵や性格，名前）できるパートナー機能や，絵と文字を使ったイメージダイアリーやイメージカレンダーがある。ほかにも約44,340語の英和辞典，約16,000語の和英辞典，約350例の英会話文例，約50,000語の漢字辞書が搭載されている。アドレス帳機能には名前，電話番号，住所のほかに似顔絵，誕生日，血液型の入力ができる。また，メモ，電卓，時計，アラームなどの機能も用意されている。

画面にはFSTN液晶（96×64ドット）が採用され，メモリ容量は32Kバイト（ユーザーエリア約28Kバイト）。

価格は，26,000円（税別）。

〈問い合わせ先〉
シャープ㈱　☎06(621)1221,043(299)8210

カセットレコーダ
TCM-AP1
ソニー

TCM-AP1

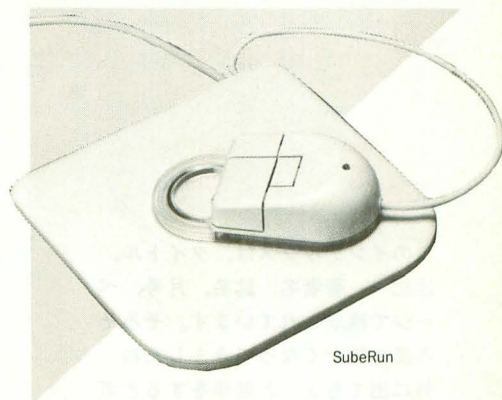
ソニーはカセットレコーダ「TCM-AP1」を発売した。

本機はテープの再生速度に合わせて自動的に音程調整を行うDPC（Digital Pitch Control）機能を搭載している。この機能は再生音声を一度RAMに落とし込み，1/16,000秒ごとにサンプリングした音をデジタル信号処理により周波数調整・加工するICを採用したことで実現した。再生モードには「オート」と「マニュアル」があり，マニュアルでは再生速度と音程を別々に調整できる。再生速度は録音した内容の＋100～－30％の範囲で設定可能。音程については，約＋1/2～－1オクターブの範囲である。

録音機能は，音声を感知したときに自動的に録音を開始するV.O.R（自動音声スタート）機能を搭載している。

価格は，22,000円（税別）。

〈問い合わせ先〉
ソニー㈱　☎03(5448)3311，06(251)5111

新感覚入力デバイス
SubeRun
フォトロン

SubeRun

フォトロンは新感覚入力デバイス「SubeRun」を発売した。

本機はデジタイザ/タブレット技術をマウスに応用したデバイスである。特徴としては，従来のマウス使用時に発生するマウスボールのミススリップをボールレスにすることにより解消した。パッドの大きさは5インチFDサイズで厚さが9mm。モニタとは1：1対応で絶対座標のタブレット感覚で操作ができる。電源は必要なく，マウスコネクタに接続する。対応している画面モードは，

640×400　640×480
1120×750　1024×768
1280×1024

の5通り。本製品はPC-9801シリーズ用のため，X680x0での使用には98バスマウスアダプタが必要である。

価格は，19,800円（税別）。

〈問い合わせ先〉
㈱フォトロン　☎03(3486)3471

FILES

Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。そろそろ風も暖かくなってきましたね。外に出てちょっと散歩をすると気分がいいですよ。ひょっとしたら春の息吹を感じるかもね。

参考文献
I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
C Magazine　ソフトバンク
テクノポリス　徳間書店
電撃王　主婦の友社
PIXEL　図形処理情報センター
POPCOM　小学館
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶特別企画　こんなゲームと遊んでみたい
ゲーム業界に関係のある人に夢のゲームを語ってもらった。――黒田幸弘ほか，コンプティーク，2月号，108-112pp.

▶NEWS COLLECTORS
高速3Dマシンと噂されるPS-Xの実像を，SCE副社長にインタビューするほかに，新音源のヤマハシンセサイザVPIの紹介，各社ペン入力マシンの比較など。――編集部，電撃王，2月号，10-15pp.

▶ここまで来た!!　CD-ROMエッチ
膨大な記憶容量で動画取り込みの世界を可能にしたCD-ROM。このCD-ROMによるアダルトソフトの実態をレポートする。――編集部，電撃王，2月号，30-44pp.

▶ゲームを継ぐもの
次世代ゲーム機の現在わかっているスペックを紹介し，開発と現場の声などから，今後主流となるハードを考察する。――編集部，電撃王，2月号，61-68pp.

▶大特集 I　'94男と女の新迷解パソコン大事典
カタカナが氾濫するパソコン業界。迷える初心者のための，正確かどうかちょっと怪しい迷解事典。――編集部，POPCOM，2月号，6-11pp.

▶NEWS CLIP
パーソナル情報ツール「Newton」，復権してきたデパートの屋上遊園地の話題などのニュースを紹介。――編集部，POPCOM，2月号，29-33pp.

▶新鮮良品館
ハイブリット自転車などハイテク製品を紹介するコーナー。――編集部，POPCOM，2月号，136-137pp.

▶周辺機器購入ガイド
新製品ラッシュが続くプリンタ。印刷方法ごとに特徴を紹介する。現行機種の一覧つき。――編集部，マイコンBASIC Magazine，2月号，44-47，205-208pp.

▶新製品 Flash NEWS
携帯性を追求したパソコン「PC-486PORTABLE」など，パソコン関連の新製品を紹介。――編集部，マイコンBASIC Magazine，2月号，60-62pp.

▶Bug太郎のプログラム・タイム
今月のテーマは「移動方法について考える」。デジタル入力，アナログ入力のフィーリングの違いをサンプルプログラムを使いながら解説。――谷裕紀彦，マイコンBASIC Magazine，2月号，68-69pp.

▶Arcade Game Graffiti
アーケードゲームの歴史を編纂する連載。第1回の今月は，ブロックくずしからスペースインベーダーの登場まで。――編集部，マイコンBASIC Magazine，2月号，145-147pp.

▶ロングラン・ヒット・ゲームズ in'93
1993年の売れ筋ソフトHOT10のまとめからロングランヒットゲームを探り，今年のゲーム界を大胆に予想する。――竹矢新，マイコンBASIC Magazine，2月号，164-167pp.

▶第9回　輝け！　日本ゲーム・ミュージック大賞
1993年下半期GMアルバム総カタログ。ノミネート作品一覧。――編集部，マイコンBASIC Magazine，2月号，172-177pp.

▶EVENT
パシフィコ横浜/会議センターで開催された「UNIX Fair'93」と東京国際見本市会場で開催された「DOS/V EXPO Tokyo」をリポート。――編集部，I/O，2月号，22-23pp.

▶Newton MessagePad
日本でも発売された「Newton MessagePad英語版」をテストしてみる。――編集部，I/O，2月号，58-59pp.

▶OS/2の世界
OS/2の歴史，プログラミングなどからOS/2を紹介する。――川元数彦，I/O，2月号，103-108pp.

▶マルチメディアの行方 2
今回は「次世代ゲーム機」を通して，家電メーカーのマルチメディアへのアプローチを考える。――奥野雅之，I/O，2月号，127-130pp.

▶特集 I　謎の次世代OSを追う
主要なOSの機能と特徴，それを取り巻く業界の動向，OSにかかわる最新マイクロプロセッサについての解説。――編集部，ASCII，2月号，185-208pp.

▶特集 2　未来への遺産 2
パソコンの現在のゲーム状況をゲームソフトの紹介から探り，今年のゲームの行方を予測する。――橋本潤ほか，ASCII，2月号，225-252pp.

▶特別企画　スペシャルインタビュー⑩
嶋正利氏に世界最初のマイクロプロセッサ4004開発にいたる経緯を聞く。――遠藤諭，ASCII，2月号，290-294pp.

▶新科学対話<2>
対談シリーズ。岩井克人氏（東京大学経済学部教授）を迎えて，経済学的な側面からコンピュータ業界を語る。――竹内郁雄，ASCII，2月号，306-312pp.

▶バカパパのモノを買い物
電動自転車やハンドマッサージャ，マウスパッドなどを紹介。――編集部，ASCII，2月号，348-349pp.

▶デジタルウォッチ
デジタルメディアやマルチメディアに関する対談時評。――桝山寛＋David D'Heilly，2月号，344-347pp.

▶未来派パソコン通信の研究<4>
今回は第1部のまとめという観点から，具体的なハードウェアを想定し，未来派パソコン通信を考える。――原田洋平，My Computer Magazine，2月号，144-147pp.

▶ビジネスマンのための情報管理術
シャープから発売されたPI-3000「ザウルス」の使用レポート。――塚田洋一，My Computer Magazine，2月号，148-151pp.

▶THE NEWS FILE
DOS/V EXPOのレポートのほか，W-VHS第1号機，全自動入浴機などの最先端機器の紹介など。――編集部，LOGIN，3号，148-155pp.

▶L.I.A.
ウィンドウズ用のカラオケソフトなど，海外からのハイテク関連の話題を掲載する。――編集部，LOGIN，3号，156-157pp.

▶HARDWARE FORUM Ver.2.0a
外付けCD-ROMドライブと，CD-ROMドライブ搭載マシンの特集。――編集部，LOGIN，3号，158-161pp.

▶第3DO帝国
3DOの日本製最新ソフト（発売予定）を紹介。――編集部，LOGIN，3号，176-177pp.

▶パソコンゲーム危機到来か？
各社の新世代ゲーム機器のポテンシャルを探る。――編集部，LOGIN，3号，182-185pp.

X1/turbo/Z

X1シリーズ

▶砂漠の人
男を操作して，苛酷な砂漠を横断するのだ！　MSX用からの移植版。――理代楓，マイコンBASIC Magazine，2月号，110-111pp.

▶ALIEN DUEL
火炎放射器でスライムを倒す。――千日前ちはる，マイコンBASIC Magazine，2月号，112-114pp.

X68000

▶SUPER SOFT EXPRESS
X68000ユーザー待望の女子高生育成シミュレーション「卒業」，「ドラゴンバスター」などを紹介。機種別，発売ソフト情報。――編集部，コンプティーク，2月号，38，43，47，48pp.

▶How to Win
「信長の野望　覇王伝」「項劉記」などのリプレイや攻略法の紹介。――編集部，コンプティーク，2月号，78-81，84-87pp.

▶年末年始ゲームオールガイド
1993年末から1994年1月にかけて発売になったゲーム72本をジャンル別に紹介する。「ネメシス'90改」「MAD STALKER X68」などが登場。――編集部，テクノポリス，2月号，6-29pp.

▶NEW GAME REPO!!
「餓狼伝説２」,「ドラゴンバスター」などの新作ソフトを紹介。機種別カレンダーつき。──編集部, テクノポリス, ２月号, 52, 53, 62pp.
▶HOT REVIEW!!
人気ゲームをゲーム業界の有名人やマンガ家に評価してもらうコーナー。今月は「ネメシス'90改」が取り上げられている。──中井覚ほか, テクノポリス, ２月号, 73p.
▶DO-JIN SOFT FAN!!
同人ソフトを紹介するページ。奥の深い対戦ゲーム「PSYCHO METAMORPHOSIS」, 1930年代を舞台にしたシューティング「LAST FORCE」などが登場。──編集部, テクノポリス, ２月号, 77, 80pp.
▶新作王
パソコン, コンシューマ機の新作ソフトを紹介。X68000用はパズルゲーム「キーパー」。──編集部, 電撃王, ２月号, 158p.
▶電撃新作予定表
X68000などの新作の発売日を一覧表で紹介。──編集部, 電撃王, ２月号, 176p.
▶HOT PRESS＋1
新作ソフトを紹介するコーナー。X68000は「卒業」「ドラゴンバスター」。──編集部, POPCOM, ２月号, 26, 27pp.
▶CPUパワーアップ工事
EPSON PC-486SEへのODP装着実体験と, CPUパワーアップサービスを行っている会社の一覧。満開製作所の「RED ZONE」も登場。──編集部, マイコンBASIC Magazine, ２月号, 56-59pp.
▶ハンマー投げ
ジョイスティックを駆使してチャレンジするスポーツゲームだ。──渋谷正徳, マイコンBASIC Magazine, ２月号, 115-117pp.
▶メイズ・ナ・ノーグ
洞窟を進み, モンスターを倒して財宝を手に入れよう。プレイのたびにマップが変わるRPG。──カイワレ, マイコンBASIC Magazine, ２月号, 118-120pp.
▶悪魔城ドラキュラ～Witched Child～
ミュージックプログラム。「悪魔城ドラキュラ」X68000版のSC-55版を参考に内蔵音源用にアレンジ。──マイコンBASIC Magazine, ２月号, 127-129pp.
▶BASIC MAGAZINE NEWS
ビデオゲームアンソロジーシリーズ第８弾「出世大相撲＆エキサイティングアワー」のレポートなど。──編集部, マイコンBASIC Magazine, ２月号, 139p.
▶X68000版「ストⅡダッシュ」プレイレポート
「ストリートファイターⅡダッシュ」のプレイレポート。──小石朋仁, マイコンBASIC Magazine, ２月号, 156p.
▶ベーマガ㊙情報局
X68000用「The World of X68000」より「ロジックラッシュ」の隠しコマンド,「ストリートファイターⅡダッシュ」用のパッドコネクタの情報を掲載。──編集部, マイコンBASIC Magazine, ２月号, 179p.
▶AV STRASSE
AVマシンの最新情報コーナー。「X Windows V11.5」「ハイパーピクセルワークス」などを紹介。──編集部, ASCII, ２月号, 315, 316pp.
▶FREE SOFTWARE INDEX
大手主要ネットにアップロードされたソフトを紹介。X68000はFAXモデムに対応したFAX送受信プログラム「faxion.x」やゲーム「マインスイーパ.X」など。──編集部, ASCII, ２月号, 383-389pp.
▶なんでもＱ＆Ａ
今月は, マニュアルに記載されていながら意外と知られていない質問に答える。──編集部, My Computer Magazine, ２月号, 170-171pp.
▶HOBBY EXPRESS
「ストリートファイターⅡダッシュ」「餓狼伝説２」などのゲームレビュー。──あゆさわかつみ ほか, My Computer Magazine, ２月号, 170-171pp.
▶NEWSOFT
「餓狼伝説２」など, 各機種用新作ソフトの内容を紹介する。──編集部, LOGIN, ３号, 14p.

▶68新聞
数多く出た落ちものパズルゲームの中で, ヒット作となった「ぷよぷよ」。いよいよX68000に登場するということで, 68新聞で特集する。──編集部, LOGIN, ３号, 168-169pp.
▶GameReview
エレクトリック・シープの「ロボットコンストラクション R.C.」を取り上げる。プログラミング魂を燃え上がらせる傑作だ。──編集部, LOGIN, ３号, 256-258pp.
▶簡単で本格的な２次元画像作成教室 1
「MATIER」の使い方のコツを紹介していく。第１回は２次元で３次元のような効果を出す方法。──長谷川一光, PIXEL, ２月号, 110, 114-119pp.

▶SX-WINDOWプログラミング
今回はイベント関係の基礎的な話。──吉野智興, C Magazine, ２月号, 141-146pp.

ポケコン

PC-E500
▶エアーホッケー
ラケットでボールを打ち返して相手のゴールを攻める対戦ゲーム。──一円ワイン, マイコンBASIC Magazine, ２月号, 122p.

新刊書案内

「超」整理法
野口悠紀雄著
中央公論社刊
新書判　232ページ
720円（税込）

この本, 売れているという。やたら売れているという。それこそ, いかに人々が「情報」の整理・管理に困っている証左というもの。情報整理術の流行も, どんどん「楽して実を取るほうへ」と変わってきた。ひとつの転機が山根一眞式（A4サイズの茶封筒にとにかく資料を放り込み, 名前をつけていく方法）ではなかったかと思う。これなら, 誰でもできそうだし, 面倒な準備はいらない。それを発展させて林晴比古氏が検索用インデックスをパソコンで管理する方法を提唱したが, これは時代に逆行していた。原資料に加えて毎回そのインデックスを打ち込む手間や, インデックスと原本が分離することによる混乱は避けられないからだ。そして,「超」整理法である。これも, 山根一眞式同様, A4サイズの茶封筒を使うものだが, ひとつだけ大きな違いがある。その封筒の管理だ。
「超」整理法では, 時間順の管理を提唱する。とにかく, 時間順に並べる。参照した資料は, いちばん新しいところに追加する。つまり, スタックに積み上げる動作と同じであり, キャッシュや辞書学習と同じ（よく見るものは決まっている。それは, 最近参照したものだ）発想だ。これは（A4サイズ封筒を並べられる場所さえあれば）非常に使いやすい。とにかく,「分類するな」「いらなくなったものを捨てやすいシステムにしろ」という発想には大拍手だ。著者は同様のシステムをパソコンのファイル管理にも適用する。ファイルは時間順に管理し（ディレクトリをジャンル別内容別に作るのではなく, 日時で作れということ。ちなみに, 私もそうしている), ややこしい検索はパソコンにまかせろ, と。これも頷ける。内容別の分類は整理に負担がかかりすぎ, 必ずどこかで破綻するからだ。
というわけで, かなり役に立つ。ただ, MS-DOSの呪縛がパソコン利用法の切れ味を鈍らせているのが残念である。（K）

「からくり」の話
中野不二男著
文藝春秋刊
☎03(3265)1211
四六判　253ページ
1,400円（税込）

「からくり」という言葉の響き, どことなく不思議で, なにか惹かれるものがある。日本のノコギリはなぜ引いて切るのだろうか？　ラムネの瓶に入っているガラス玉はどうやって入れるのだろうか？　疑問というほどおおげさではないが, そのからくりを知っていると楽しいものである。
本書は, からくりをめぐる古今東西の技術に関する雑話である。著者は１つひとつのからくりを通して, 日本の技巧と西洋の技術との違いを感じとっている。ただ, そんなことを抜きにしても, 紀元前から現代までの技術に関するいろんなこと, 意外なことに興味をそそられる１冊である。

マルチメディアマインド
浜野保樹著
ビー・エヌ・エヌ刊
☎03(3238)1622
四六判　349ページ
2,000円（税込）

本書は, マルチメディアについてテクノロジーからのアプローチではなく, 人間や社会の側面から語っている。具体的には, 日米の映画業界やアメリカの情報戦略などについてである。
著者はマルチメディアのもっとも単純な定義を,「情報をデジタル化するツール」としている。アナログの時代における映像の制作は, マスメディアのものであった。そしていま情報がデジタル化されることでパソコンというパーソナルなメディアで扱えるようになった。
つまりマルチメディアは, 映像などの表現の主体を個人へと移行していく可能性をもっている。

XF1キーを押すと強制的に停止するプログラムを作ろうとしたのですが，うまくいきません。一応，止まることは止まるのですが，キーボードからの入力を受けつけなくなります。どうすればいいのでしょうか。

愛媛県　大森　亮寛

リスト1が大森さんが書いたプログラムです。まずこのプログラムの流れを順に追っていきましょう。

プログラムを実行するといきなりキー入力割り込みベクタをcheckkeyに書き換えて常駐終了するようになっています。これでキー入力割り込みがあるたびにcheckkey～に制御が移るようになります。

checkkeyではレジスタを退避したあと，IOCSコールを使ってXF1キーの状態を調べます。もしもXF1キーが押されていなければ，退避したレジスタを復帰して本来のキー入力割り込み処理に制御を戻しますが，XF1キーが押されているならstopadd～の強制終了処理に制御を移します。stopaddではDOSコールで画面クリアしたあと，終了コード1で終了するようになっています。

大森さんのプログラムでいくつか気になった点を修正したのがリスト2です。最初にキー入力割り込みベクタの設定です。大森さんはキー入力割り込みベクタをIOCSコールのB_LPOKEで変更していますが，どうせIOCSコールを使うのなら，例外処理，IOCSコールのベクタを設定するB_INTVCSを使うほうがいいでしょう。プログラマーズマニュアルによると，キー入力割り込みのベクタ番号は$4Cだということですので，プログラムは，

```
    moveq.l #$80,d0
    moveq.l #$4c,d1
    lea.l   checkkey,a1
    trap    #15
```

などのようになります。

あと，これは私の場合ですが，ベクタを設定するときには割り込みを禁止するようにしています。

次に割り込み処理部分に移ります。例外割り込みの中で，さらに例外割り込みを発生させるなんて気持ち悪いことはやめましょう。大森さんのプログラムではIOCSコールの_BITSNSを使ってXF1キーの状態を調べていますが，これをやめて別の方法を考えてみます。

X68000とキーボードはMFPのUSART（シリアルポート）を介してデータの入出力を行っています。キーが押されたり離されたりするとUSARTデータレジスタ($E8802F)にキーコードが格納され，キー入力割り込みが発生しますので，割り込み処理では$E8802Fの内容を調べることにします。

XF1キーのキーコードは55ですから，$E8802Fの内容が55ならstopaddへ飛ばすようにします。55以外なら正規のキー入力割り込み処理アドレスにジャンプさせます。大森さんはEXPERTに内蔵されているROMをもとに絶対アドレスを指定していますが，ROMバージョンの違う機種でも動作させたいのなら，ここはB_INTVCSの戻り値がd0レジスタに元のベクタアドレスを格納しますので，これを利用するとよいでしょう。リスト2ではお行儀の悪いことに自己書き換えをしています。割り込み処理中でレジスタの値を破壊できないために，このような形をとりました。

stopadd～は当初大森さんのプログラムと同じにしていましたが，強制終了したあと再びXF1キーを押したら暴走してしまうので少し変更しました（理由はわかると思います）。キー入力割り込みを禁止したあと，キー入力割り込みベクタアドレスを元のベクタアドレスに戻す処理を加えました。IOCSコールを使わないために直接$130番地に書き込んでいます。

ということで，大森さんのプログラムの悪い部分はキー入力割り込み処理のcheck

リスト1

```
        .text

startadd:
        move.l  #checkkey,d1
        move.l  #$130,a1
        moveq   #$88,d0
        trap    #15
        clr.w   -(sp)
        move.l  #endadd-startadd+1,-(sp)
        .dc.w   $FF31
checkkey:
        move.l  d0,buf0
        move.l  d1,buf1
        move.w  #$0a,d1
        moveq   #$04,d0
        trap    #15
        cmp.b   #32,d0
        beq     stopadd
        move.l  buf0,d0
        move.l  buf1,d1
        jmp     $FF15EA
stopadd:
        move.w  #2,-(sp)
        move.w  #10,-(sp)
        .dc.w   $FF23
        addq.l  #4,sp
        move.w  #1,-(sp)
        .dc.w   $FF4C
buf0:   .ds.l   1
buf1:   .ds.l   1
endadd:

        .end
```

リスト2

```
        .include        iocscall.mac
        .include        doscall.mac

        .text

startadd:
        clr.w   -(sp)
        DOS     _SUPER
        move.l  d0,(sp)
        ori.w   #$700,sr                        * 割り込み禁止
        pea.l   checkkey(pc)
        move.w  #$4c,-(sp)
        DOS     _INTVCS
        addq.l  #4,sp
        move.l  d0,orig4c_adrs
        andi.w  #$f8ff,sr                       * 割り込み許可
        DOS     _SUPER
        addq.l  #4,sp

        clr.w   -(sp)
        move.l  #endadd-startadd+1,-(sp)
        .dc.w   $FF31
checkkey:
        cmpi.b  #$55,$e8802f
        beq     stopadd
        .dc.w   $4ef9                           * JMP
orig4c_adrs:
        .ds.l   1
stopadd:
        bclr.b  #4,$e88013                      * キー入力割り込みマスク
        move.l  orig4c_adrs(pc),$130.w          * キー入力割り込みベクタを元に戻す
        bset.b  #4,$e88013                      * キー入力割り込みマスク解除
        move.w  #2,-(sp)
        move.w  #10,-(sp)
        DOS     _CONCTRL
        addq.l  #4,sp
        move.w  #1,-(sp)
        DOS     _EXIT2

endadd:

        .end
```

keyだということになります。どこが悪かったのか説明しましょう。

まずXF1キーの状態を_BITSNSを使って調べています。_BITSNSはIOCSコールのキー入力関係のワークエリアの内容を調べて戻り値を設定しています。よく考えればわかるでしょうが，XF1キーが押されてcheckkeyが呼び出された時点では，IOCSコールのワークエリアにはXF1キーが押された状態は書き込まれていません。このあと正規のキー入力割り込み処理で初めてワークにXF1キーが押されたことを示す情報が格納されます。ですから，正確には大森さんのプログラムはXF1キーが押されてから2回目のキー入力割り込み処理が発生したときにstopaddに分岐するというものなのです。

最後に致命的な問題として，キー入力を受け付けなくなる原因ですが，これはUSARTデータレジスタの内容を読み出していないためです。キーボードの内蔵CPUは本体が前のデータを受け取ったことを確認してから次のデータを送るようになっています。キーボード側から見れば送信バッファ内にあるキーコードを本体が受信（読み出し動作を）してくれないので，いつまでたっても送信バッファが空にならず，次のキーコードを送信できない状態になっているのです。

ここまで引っ張っておいていうのも意地悪ですが，大森さんのプログラムのcheckkeyの先頭部分になんでもいいですから$E8802Fの内容を読み出すような命令を書いておけば，停止後もキーボードの入力を受け付けなくなるようなことはなくなるはずです。

ただしstopaddに分岐して強制終了したあと，最初のキー入力割り込み処理の時点ではワークにXF1キーが押された状態が格納されていますので，どのキーを押しても再び強制終了処理が実行され暴走してしまいます。（朝倉　祐二）

1994年1月号のストリートファイターIIダッシュの紹介記事のなかでマルチAD PCMドライバについて書いてある部分に，括弧して「X68000でも一応使えるようだが」とありますが，具体的にどのようにすれば使うことができるのでしょうか。

神奈川県　志田　潤一

ストリートファイターIIダッシュのADPCM4.XをX68000で使うにはどうすればいいのでしょうか？

また，餓狼伝説2にもPCM8を組み込むことはできるのでしょうか？　いずれも詳しく教えてください。東京都　秀平　良忠

簡単なので解説はしなかったのですが，あまりに問い合わせが多いのでここでお答えします。

まず，ストリートファイターIIダッシュのAD PCM多重化ですが，AUTOEXEC.BATを見ればわかるように，実行しているマシンのCPUの種類に応じて起動するAD PCMドライバを切り換えています。

ハードディスクをお使いの方は起動バッチファイルのなかの，

ADPCM%VOICE%

の部分を，

ADPCM4

に書き換えて実行するだけで大丈夫です。

フロッピーディスクでお使い方は，ストリートファイターIIダッシュのシステムディスクのバックアップをとり，同様の変更を加えてください（DISKCOPY.XでOK)。

起動方法です。ここで作成したディスクをドライブ1に入れてX68000を起動してください。ドライブ1を読みにいったらドライブ0にマスターのシステムディスクを入れておきます。あとは画面の指示どおりにディスクを入れ換えてやれば大丈夫です。

せっかくですから起動バッチファイルについて少し解説しておきましょう。

まず，MPUTYPE.Xを実行し，CPUの種類を調べています。Human68kver.3.0で拡張されたIOCSコールを使っているので68000から68040にまで対応しているはずです。で，68030だったら環境変換VOICEに4をセットします。ちなみにVOICEにはあらかじめ1がセットされています。これをADPCMという文字列と組み合わせてコマンド名を作り，実行しているわけです。68000ならADPCM1.X，68030ならADPCM4.Xが常駐することになります。

ストリートファイターIIダッシュはデフォルト（特別な指示をしていない状態）ではX68030でのみ音声多重に対応するようになっていますが，X68000XVIくらいであればADPCM4.Xを使ってもほぼ遜色ない動作が可能です。10MHz機ではかなり動作が重くなりますが，これはしかたないことでしょう。

餓狼伝説2についても同様な方法で起動すればAD PCMを多重化することができます（もちろんあらかじめバックアップしておいたディスクにPCM8.Xを転送しておく必要がありますが)。

AUTOEXEC.BAT中でZMSCの起動オプションに〝－O4〟などの指定を加えておいてください。しかし，餓狼伝説2の場合は一部多重化しないところがあり，完全な音声の多重化はできないようです。

これらのゲームでは，ユーザーの手でこういった変更が行われることが，かなり意識された作りになっているようにも思われます。ただし，正式なアナウンスがないということは動作保証をしないということですから，これらの件でソフトハウスに問い合わせることは絶対に控えてください。

今回は記事中で触れていたこともあって解説しましたが，基本的にゲームの改造などに関する質問にはお答えしない方針になっているので，今後同種の質問はご遠慮くださいますようお願いします。

また，今回の解説を参考にしてもまだよくわからない人や，AUTOEXEC.BATを読んでもよくわからないという人は，そもそもこういうことに手を出してはいけないんだと思っておいてください。（中野　修一）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してください。

宛先：〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

「2月は逃げて走る」というように，もう3月です。春風も吹いてきそうな今日この頃，進学や卒業のシーズンですが無事に進路は決まりましたでしょうか？ 生活に変化のある人もない人も，体に気をつけてがんばってくださいね。

◆とうとう（というかやっと）出ましたね。Z-MUSICシステムver.2.0！ ver.1.0のとき買いそびれてしまい，それ以来このver2.0が出るまでのあいだかなり待ちましたよ。システムだけ配布されてもPCMがなきゃなんの意味もないですからね……。でもようやく苦労して手に入れたのだからじっくりと使っていくつもりです。これからも特集組んでください。

堺　和幸(20)宮城県

なんとか1月号の発売にver.2.0の発売が間に合ってひと安心。いままでZ-MUSICを知らなかった人も1月号の特集をきっかけに使ってくれると嬉しいです。

◆新連載「"実戦！" ゲーム作りのKNOW HOW」に，とても感謝します。こういったゲームのプログラミングテクニック関係の記事をずっと待っていた自分としては嬉しいかぎりです。協力に「H.C.S.」とありましたが，これは「ヒューマンクリエイティブスクール」のことですか？ 実は僕，ここに入学する予定なんです。

長田　良太(18)神奈川県

◆ぜひやってほしいと思っていた，アセンブラレベルのゲーム作成講座が「"実戦！"ゲーム作りのKNOW HOW」として始まったのでとても嬉しい。私もアセンブラでプログラムをするようになってまだ1年足らずのアマチュアプログラマなので，非常に興味がもてます。現在シューティングゲームを作っているので，スプライト関係のテクニックについてやってもらいたいと思います。

千葉　浩貴(21)宮城県

新しい連載は好評のようです。やはりアセンブラでプログラムを組みたいという方は多いみたいですね。

◆1月号，DōGAの森山効果（アニエフェ）は非常に参考になりました。実は，とある発表会のときに未熟なピカピカ効果をやったんですけど，それが見事に悪い例だったのは情けなくなってきました。記事を見る前に，真っ白にするというのは知っていましたが，なにも描かないというのは意外でした。次回作はもっとすごいものを作りたくなりました。

片平　正二(19)神奈川県

記事のなかにもありましたけど，実際に自分で試行錯誤していくことで自分の力となっていきます。次回作はきっと前よりも素晴らしいものができると思いますよ。

◆「こちらシステムX探偵事務所」や「ハードコア3Dエクスタシー」は，高度なプログラムを現在進行形で作ってよく連載ができるもんだと感心してしまいます。X68000を持っていないから実際に動かせないけど，今年は就職なので新機種を買ってやるもん！ と決意している私でした。また「猫とコンピュータ」のトオルに共感した私の愛読書は『坊ちゃん』です。人間まっすぐ生きたいものです。

福田　強(19)東京都

新しい機種が早く出るといいですね。噂だけはたくさんあるんですけどね。

◆3DOマルチプレイヤー「REAL」を触ってみた。ただ，ソフトがサンプルCD＆バンドルの「CRASH'N BURN」というレースゲームのみであまりスゴイと思わなかった。やっぱりコンピュータはハードとソフトのバランスだな。

遠藤　勝博(23)宮城県

日本での発売も3月20日とあと1カ月くらいですが，新しいソフトは増えているんでしょうか。期待半分，心配半分といったところですね。

◆「餓狼伝説2」は予想以上にいい出来だ。某格闘ゲーム「スト○'」と比べてもあらゆる点で勝っていると思うのだが，誌上では「ス○II'」有利といった雰囲気である。誰か私の味方になってくれる人はいないだろうか。

田中　康治(22)東京都

4月号の「GAME OF THE YEAR」の発表でどうなるか楽しみですね。

◆「microOdyssey」で「HiFi，HGなど，テープで画質が違う！」というのを見てびっくりしました。私は標準と3倍の画質の違いすらわからないのに……。気合が足りないのでしょーか？ ちなみにCDの音質の違いというのも全然識別できません。金がかからなくていいですけどね。

樋口　泉(19)福岡県

そう，気合がたりーん（嘘）。

◆卒業にむけて忙しい日々をおくっています。年末のバイトに卒論，そして部活，4回生がこんなにハードな生活をおくっていいのだろうかと考えてしまう，今日この頃……。とほほ，もしかして5回生なんてことは……いや考えたくない。

坂井　大吾(22)兵庫県

そろそろ4年で終了か5年目に突入か決まった頃でしょうか？ とりあえず，きっと卒業が決まったと信じて，「おめでとうございます」といっておきますね。もし違ったら……。

◆1993年12月21日をもって退職してしまいました。これから1年くらいアルバイトをしながら自分のやりたいことをしていきたいと思います。

小川　毅(20)埼玉県

やりたいことがあるというのは素晴らしいことだと思うのでがんばってください。

◆せっかくハードディスクを買ったのに動いてくれない……。まあ一応アクセスランプがてかてかと点滅してるからつながっているとは思うけど，なぜか使用不可。あぁ，シリィちゃん（X68000PROの名前，元ネタバレバレ？）オレがなにをした？ メモリも4Mバイト積んだし，

◀岡村　直也　兵庫県
知らぬまにお姉さんがX68000を使いこなしています。これはお義兄さんの……やはり愛の力か？ 残ったローンはお祝いってところですね。

◀小川　伸輔　宮城県
棚に並んだOh!X，いつの号から並んでいたのでしょう。仙台に住んでいるX68000ユーザーでバックナンバーのほしい人は急ぎましょう。

ハードディスクもつなげたのに……。で，しかたなくシャープに電話したら「もしかして5,000円とか6,000円とかのケーブルつなげていませんか？」「……もっと安いやつです」「……」。どーやら安いケーブルだとだめらしい。こーして私は雪道のなか，1時間かけてハードディスクを買った店へ行くのであった。

松本　勝正(20)富山県

このハガキが届いたということは，雪道のなかで倒れたわけではないですね。でも戻って来たとも書いてないから，霊魂だけがほかの体にとりついてこのハガキを……。

◆うちの大学の学食では食券の販売機のなかにX1turboZが生息している（モニタつき）。

阿部　恒(20)神奈川県

今度は，いかにして生息しているかのレポートをお待ちしております。

◆最近はいろいろな本を買うのをやめて，図書館へかよっている。そうしただけで，だいぶ金がたまってきた。しかしながら，Oh!Xは買い続けている。不思議だ。なぜかやめられない？

川田　宏(19)香川県

ありがたいことです。Oh!Xはパソコン誌のかっぱえびせんになれるのだろうか。

◆大掃除をした。もちろんX68000の周りもこれでもかというほどキレイにした。デスクを移動させたとき大量（といっても5，6枚）のフロッピーが出てきた。以前にもこんな記事が載っていたけど，まさか本当になくして忘れているとは思わなかった。綿ボコリに包まれていたフロッピー。怖くてドライブに入れられない。いったいなかには，なにが入っていたんだろう？

路川　圭一(18)茨城県

中身を忘れてしまうくらいだからきっと大事なものは入ってない……といいですね。ひとごととは思えないんですけど……。

◆先日，友人宅の猟犬がイノシシにアゴの骨を砕かれた（実話）。　藤田　康一(23)静岡県

成年だというのについてない犬ですね。

◆SEGAのゲームセンターで「バーチャファイター」を見ました。思わず映画「トロン」のゲーム世界に閉じ込められた主人公を思い出してしまいました。なんだか怖くて夜も眠れずって感じです。特にあの変にリアルな腰の関節は気持ちが悪いです。アンバランスな顔もちょっと怖いし……。そのうち服が乱れたり，靴が脱げたり，流血したりするんでしょうか。ゲームセンターへ行くのが怖くなりそうです。ところで入試の前々日だというのにOh!Xの発売日という理由だけで喜々として書店の自動ドアをくぐってしまった僕は，本当に2浪せずにすむのでしょうか。ちなみにいまは前日の夜です。

平　学(19)東京都

こんなハガキを書いている余裕があれば大丈夫……かな。

◆自分で自分の車を壊すのはよいが，酔っ払いに車を壊されるのはとても腹が立つ。Civic vtiのボンネットがボロボロ（悲）。

八亀　圭一(19)神奈川県

◀岩瀬　貴代美　福岡県
そろそろ卒業シーズンですが，無事(?)卒論が終わって，ネメシスをプレイしているんでしょうか。ただ，くれぐれも体は大切にね。

◀橋本　和典　東京都
このチュン・リーはずいぶんケバくないですか。アイシャドーもバッチリてな感じでこのセリフ，きっとタカビーにちがいない。

今度その酔っ払いを見つけたら近くの電柱にでも縛っておきましょう（冗談）。

◆酒の飲める体質か飲めない体質かを調べるパッチテストというものをやってみました。結果は大酒飲みタイプということでした。忘年会，新年会でも安心して飲めそうです。

齋藤　眞二(20)東京都

いまごろは大酒飲んでまだ酔っ払っているところでしょうか？　あ，当然してないと思いますが，人の車にはイタズラしないようにしましょうね。

◆青春18きっぷで帰省（京都→千葉）すると静岡県がいかに広いかがよくわかる。

松永　正弘(23)京都府

時刻表を手に静岡県の東海道線の駅の数を調べると38駅。普通列車で横断すると……広いですね。

◆やったー！　卒論が出せた。締め切り5分前に書き上がり，終わってみれば752番目のブービーでありました。しかし卒論ってすごいですね。友人はもちろん，普段あまりつき合いのない人にまで手伝ってもらったのです（感謝のしようがない。走ってもらった人もいますし……）。てなわけで，友人への多大な貸しはあとかたもなく消えてしまったのでした。しくしく……。

中島　太郎(22)神奈川県

友人への多大な貸しがなくなっても，卒論が出せたのだからいいじゃないですか。最後に提出した人もたくさんの友人に手伝ってもらったのでしょうか。ただ，友人に手伝ってもらっても締め切りに間に合わなかった人は……いるんでしょうね。

◆金があると暇がない。暇があると金がない。どっちもほしい……。　森山　茂雄(19)熊本県

悩みは同じですね。

◆右手人差し指の骨を折ってしまったー。仕事中のことで，折ったときはあまり痛くなかったけど，これを書いているいまはとっても痛いっす。人を殴るときは手をしっかり握らないと折っちゃうよ。さて，私の仕事はなーんだ（答えは来月号で）。　片山　晃一(25)栃木県

人を殴る職業なんて……ボクサーですか？でもハガキの表を見ると違うみたいだし，まさかヤのつく職業では……。

◆最近，物忘れがひどくてハガキを書いてもずっと出すのを忘れていた……。

黒岩　智教(23)大阪府

忘れたハガキはどこにいったんですか？

◆今年のX'masも残業だった。去年もその前もなにか仕事していたような……。やっぱりなにかが間違っている！　内藤　陽一(27)東京都

仲間ですね。間違っているとは思ってもまた今年も……なのかなあ。

◆第2種情報処理試験に合格しました。これで安心して就職浪人できそうです（シクシク）。ところで通産大臣より科学技術庁長官のほうがカッコイイと思いません？程田　勝也(20)茨城県

今年は景気がよくなりますように……。X68000の周りも景気がよくなるといいんですけど……。

◆睡眠時間がほしい……。

来島　克樹(21)広島県

お休みなさい。

◆いま，思いもしなかった病気（気胸）で入院しています。やりたいことはいくらでもあるのにとても暇です。ついに究極の筆無精の私もハガキにペンを走らせています。あー，部屋の掃除が，ファイルの整理が，制作途中のプログラムが，ストIIが……。　安井　建史(21)奈良県

くれぐれも早い回復をお祈りしてます。せめて病院にパソコンが持ち込めれば……やっぱり無理ですよね。

◆女をくどくときは歌で落としましょう。歌って見つめればバッチリです。

大川　輝比古(19)茨城県

恥ずかしくてとてもそんなことできません。あとオンチな人はどうすればいいですか。

◆X68000初代でZVT.Xの畳み込み処理を行ったら……終わらない。佐怒賀　英一(26)神奈川県

そろそろ終わりましたでしょうか？

◆岡村さんってお金持ちなんですね（毎月なにかしら買っている）。　八尾　唯人(17)神奈川県

1993年1月から調べてみるとハード2台（スーファミ含む），周辺機器3つ，ソフト3本という結果でした。さて以上のものを買える人はお金持ち？

◆小森まなみのX'masアルバム「Noel」を聴いていると忘れていた「なにか」を思い出させてくれます。特に「童話－小さな雪の物語」は思わず涙がウルウルと……。市川　徳明(20)東京都

忘れていた「なにか」を知りたい人は聴いてみるといいかもしれませんね。

◆学校からの帰り道，ほとんど毎日のように僕の前を横切ってゆく子猫がいる。飼ってやりたいけど，家を空けることが多いし……。

小山　優一(20)東京都

その猫はいまも小山さんの前を横切っていくのでしょうか？　それとも別の飼い主が現れて……。

◆手持ちのお金が心細かったのでほかの本と一緒にクレジットカードでOh!Xを買ってしまいました。こんなこといつまで続くんだろう，ううう……。　　松尾　美千代(26)大阪府

来月もクレジットカードで買った余分な本で手持ちのお金が心細くなって……いつまでたっても苦しいような気が……。

◆今日は冬至です。外は雪が薄く積もってます。こんなときにゆずの入っているお風呂に入ると，日本っていいなあとしみじみ思います。

木村　奨(21)兵庫県

温かいお風呂にゆっくりと漬かって，お酒でもキューッと……。

◆数年前，本屋さんで立ち読みしていたら25歳くらいの兄ちゃんがやってきて「この人の描くマンガすごくおもしろいよな，すごいよなあ」とやたら誉めまくって去っていったことがあった。会社で私はX68000を誉めまくる……あの人と同じだ。　　伊福　透(23)沖縄県

みんな誉めるものが好きだからこそということで……本当にいいかどうかは人によって……。

◆1994年の年賀ハガキで当たるお年玉切手シートは41円だろうか50円だろうか。41円切手だと15日の当選発表から1週間ほどしかそのままで使えない。どうなるのだろう。

伊南　尚幸(18)青森県

切手シートはなんと41円と62円切手でした。見事に裏をかかれましたね。でも，こんなのもらってどうするんでしょう……。郵政省はなにを考えてるんだろう。

◆プレゼントの「悪魔城ドラキュラ」が届き，親子3人楽しく遊んでいます。5面には苦しみましたが，なんとか最終面までクリアすることができました。年甲斐もなく熱中してしまいました。しばらく休んで6面のボスとまた戦いたいと思います。手抜きのない素晴らしいソフトでした。ありがとうございました。

井門　清(43)愛媛県

十分に楽しんでいただけたみたいで，プレゼントのかいがあります。しかし年のわりには（ごめんなさい）やりますね。

◆やっぱり本当に好きなものは手に入れるのではなくて見ているほうがいいのかもしれない。私の購入したパソコンは次々と……いうまい。別に好きな人の愛を得られなかったなど……いうまい。どうすればいいんだか？

中村　学(21)福岡県

本当に好きならやはり手に入れたいと思うのでは……。

◆サイフのなかには300円もない。「ストII'」や「餓狼伝説2」が福沢先生をもっていってしまった。おかげで「Z-MUSICシステムver.2.0」は来年に持ち越しだ。ついでに時間までもっていかれると卒論が書けないぞ。え，書かなきゃかえって時間ができるんじゃないかって……そんなあ。　　小林　裕昭(24)東京都

ひとりツッコミひとりボケ，ご苦労さま。

◆バイト先へ車で行く途中，左カーブにさしかかった瞬間，運転席側のドアが全開しました。身を乗り出して閉めましたが，うしろの車やバイクは怖がって近づいてきませんでした。とりあえず対向車が来てなくてよかった……。ちなみに，助手席にいた友人は寝ていて気づかなかったそうです。幸せな奴，50km/hは出てたから対向車がいたらかすり傷ではすまなかったぞ。

今田　智宣(19)兵庫県

開いたドアが助手席側だったら，どうなっていたのだろう。寝ていた友人が起きたときはうしろの車が目の前に……。

◆地方公務員になる私にとって最大のメリットは，転勤が市内に限られるということにつきます。まあ，運悪く東京という可能性もありますが新人は大丈夫でしょう。1月号の町田さんはクビにならないということを書いておられましたが，交通事故(重度)でも起こそうものなら，まず内定取消でしょう。新井　誠治(22)北海道

前を走っている車を追い越そうとして突然開いたドアにぶつかったりしないでくださいね。世のなかには，そんなこともあるみたいですから……。

◆久しぶりにタケルを覗いたところ，「年賀状イラスト集 成年編」というのがあって一瞬びびったが，よく見ると「戌年」だった。

要　秀紀(21)京都府

実際に成年編があったら，内容はやはりムフフなのでしょうか。

◆近くの書店で「Z-MUSIC」を注文した。説明すると長くなると思い，Oh!Xの広告ページを開いて「これ注文してください」とお願いした。店員は「わかりました」といって，広告を注文書に写していた。それにしてもやはりマイナーなのかな。マイナーなものを注文するときは，なにであれ実物がわかるものを見せて「これください」がいちばんわかりやすいと思う。また親切であろうと実感しました。

壁谷　善嗣(34)愛知県

なにはともあれ，ご注文いただきありがとうございました。

◆ひとり暮らしは一生に一度はすべきです。生活費のやりくりはまさにシミュレーションゲームです。母の強さが身にしみます。でもまだ親のスネかじりですからマシなのでしょうけど。

間宮　義晴(18)山形県

そういえばここのところ家計が苦しいような……だからシミュレーションゲームが苦手なのかなあ。

◆少し前に“ボウリングのオートスコアを見ているとロジックが頭に浮かぶ”で登場したプログラマです。最近は症状が悪化して寝言で“4重ループ，アドレス加算”などと叫び家族を気味悪がらせています。　小倉　圭司(24)東京都

そのうち寝ながらキーボードを叩き始めたり，コードを口走ったり……。家から追い出されないように気をつけてくださいね。

◆不況のおり，会社をクビにならないように会社のパソコンでは遊ばないようにしている今日この頃です。　　石川　栄一(35)新潟県

不況のおり，会社をクビにならないように会社のパソコンでまじめに遊んでいます。

◆タバコには製造段階で，トップフレーバーと呼ばれる香料がスプレーされます。香料はファミリーごとに違い，マイルドセブン系にはチョコレート，キャスター系にはバニラ，ハイライト系にはラム(洋酒)，セブンスター系にはスパイスもの，と独自なものになっています。マイルドセブンは好きだがキャスターはちょっと……という人はこの香料が合わないのでしょう。

市川　勝彦(22)神奈川県

どのタバコも合わない人はタバコのなにがだめなんでしょうね。

◆E.K.さんへ。「リーサルエンフォーサーズ」(コ

▲大高　孝平　宮城県
初投稿で初掲載，ラッキーですね。今月はカラーイラストの応募が多くてこのコーナーのイラストが少なかったのです。次もがんばってください。

▲江副　滋　埼玉県
バスでスキーに行く人が減っているみたいです。車で行く人が増えたのでしょうか。イラストではバスのなかの暖かさが出ててグッドですね。

ナミ）の警官の「うっちゃだめだ（早口で）」はどうですか？　本当は英語でなにかいってるんだろうけど……「お待たせ」もある（銃を拾ったとき）。　河野　裕文(18)静岡県

◆ゲームの空耳といえば春麗の「スピニング……」が「イチ，ニ，サン，シ」と聴こえる人もいるようです。かくいう私は「波動拳」が昔「いや～んケン」に聴こえてました（効果音とかぶさってるんだもん）。あと古いところで，「源平討魔伝」の義経のセリフが「かかってこんかい」と聴こえるといってゆずらない友人がいます。別の友人は「通してしんぜよう」とかいうし……通してどうすんだ。
山下　智也(23)大阪府

もとのセリフはなんていってるんだろう。知ってる人がいたら教えてください。

◆去年1年を振り返ってみると1：1：5：5＝岡山：山口：埼玉：沖縄で，その県にいたかな。それにしても忙しい1年だった。P.S.東丈の「タイガーキック」は「タイヤキーッ」と聴こえる。　藤原　彰人(23)岡山県

移住の原因は，やはり仕事関係なのでしょうか。今年もいろんな県で忙しい日々をおくられるんですか？

◆私は見た。教習所でAT限定にもかかわらず第1段階で3回も乗っているおばさんを！
渡辺　幹司(21)愛知県

◆1月号の手嶋さんへ。私の行った教習所では過去に緊張して力が入りすぎ，方向指示器のレバーを折ってしまった人や，ついにはシフトレバーまで折った人がいたとかいないとか。ちなみにこれが載れば，教習所ネタのリレーが私で4人目です。　藤田　敬三(18)山口県

これで上の渡辺さんを含めて5人目ですね。それにしてもレバーを折ってしまうとは……でもそんな人が案外細身の女性だったりするんですよね。

◆やはりバイトしなきゃダメらしい。
岩崎　幹(18)東京都

がんばって稼いでください。

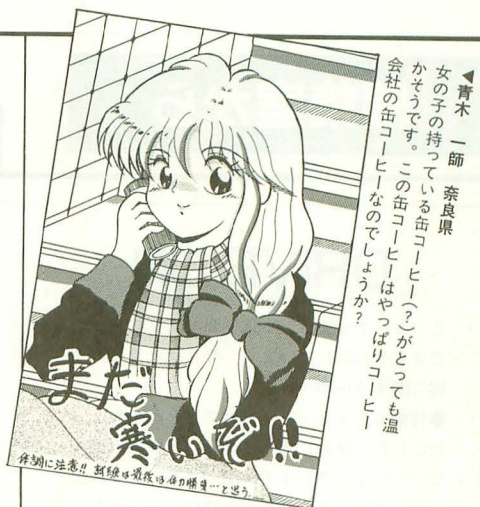
◆いつも疑問に思っていたんですが，ここ（編集部へのメッセージ）に書くと，どこに載るのでしょうか？　いつもドキドキしてOh!Xを読むのですが，まさかどこにも載らなかったりして。
長谷川　祐之(16)新潟県

ということでここに載ります。ほかには，ページの下部分に載ることもあります。これからもよろしくお願いします。

◆いつもどおりOh!Xを買って帰り，さあ読むぞと表紙を見たら，1本の長い折れ目が入っていた。これからは，上から3冊目の表紙を見て買おうと思った。　前田　基行(18)兵庫県

なかなか探すのが大変というハガキもいただきますので，選べる本があるだけ幸せかもしれません。

◆今年もXの世界に幸多からんことを。S-PULSEの優勝はだまってても転がり込んでくる……といいなあ。　亀田　徳隆(18)香川県

読者のみなさんに暖かな春が訪れ，素晴らしい1年でありますように。

◀青木一師　奈良県
女の子の持っている缶コーヒー(?)がとっても温かそうです。この缶コーヒーはやっぱり会社の缶コーヒーなのでしょうか？

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目（売る・買う・氏名・年齢・連絡方法……）を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

売ります

★アイ・オー・データ機器の1Mバイト内蔵増設RAMボード「PIO-6BE1-AE」（X68000ACE/PRO/PROII用）を5,000円（送料込み）で売ります。新品同様です。連絡は往復ハガキでお願いします。〒606　京都府京都市左京区岩倉中在地町24-15　寺田　まさよし(21)

★アクセスのX68000用DOSエミュレータ「CONCERTO-X68K」を25,000円で売ります。連絡は官製ハガキでお願いします。〒167　東京都杉並区上井草2-26-9　秋山　和徳(22)

★アイテックのハードディスクドライブ「TX80」を40,000円で売ります。色は黒で，箱なし，取扱説明書，ケーブル，ターミネータはあります。また，24ピンカラー漢字プリンタ「CZ-8PG1」（本体のみ）を30,000円で売ります。連絡は往復ハガキでお願いします。〒343　埼玉県越谷市弥栄町1-105-27　加茂　拓也(19)

★X68000用数値演算プロセッサ「CZ-6BP1」を30,000円，アイ・オー・データ機器の2Mバイト増設RAMボード「PIO-6BE2-2M」を4Mに増設したものを30,000円で売ります。どちらとも付属品すべてありで完動品です。往復ハガキで連絡をお願いします。〒631　奈良県奈良市千代ヶ丘1-9-13　桜井　秀一(25)

★X68000用4Mバイト増設RAMボード「CZ-6BE4C」を35,000円で売ります。連絡は希望価格を明記のうえ，往復ハガキでお願いします。〒820　福岡県飯塚市下三緒35-620　内藤　大祐(28)

★システムサコムのX68000用MIDIボード「SX-68M」＋ローランドのサウンドモジュール「CM-32L」を30,000円で売ります。箱はないですが，説明書はあります。連絡は往復ハガキでお願いします。〒039-23　青森県上北郡六戸町犬落瀬若宮51　杉山　直樹(21)

★HAL研究所のハンディスキャナ「HGS-68」を15,000円（送料別）で売ります。箱，説明書，付属品すべてあります。連絡は官製ハガキでお願いします。〒538　大阪府大阪市鶴見区諸口1-14-11-408　西川　静夫(39)

買います

★シャープのMIDIボード「CZ-6BM1A」かシステムサコムの「SX-68MII」と，ローランドの「SC-55」か「SC-55mkII」をセットで40,000円で買います。説明書，付属品があるものでお願いします。連絡は往復ハガキでお願いします。〒480-11　愛知県愛知郡長久手町長湫丸根22-5　森田　僚(18)

★MZ-2500用ハードディスクインタフェイスボード「MZ-1E30」もしくはテレシステムズの「MZ-251」を25,000円で買います。完動品でマニュアルとユーティリティソフトがあれば箱なしでもかまいません。連絡は官製ハガキでお願いします。〒270　千葉県松戸市六高台3-106 パルーム六高台206号室　遠藤　憲(23)

★400ライン・デジタルRGB対応15型カラーディスプレイテレビ「CZ-870D」以降のシリーズを30,000～48,000円（送料込み）で買います。アナログの有無や状態によって価格に＋αもします。連絡は往復ハガキでお願いします。〒837　福岡県大牟田市大字草木256　塚本　尚伸(27)

バックナンバー

★Oh!X1993年3月号を送料別で3,000円にて買います。切り抜き不可。連絡は官製ハガキでお願いします。〒382　長野県須坂市須坂1230-43-706　阿川　良輔(25)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は1月号の内容に関するレポートです。

●特集は，マニュアルだけではわからなかったことが，簡単に書いてあり，役立つものになっていると思います。が，「Z-MUSICシステム」は機能が多く初心者には使いこなせない部分も多々あると思います。そういう私も，昔は自分で譜面を見てよく曲を打ち込んでいたのですが，最近はソフトの進化にとてもついていけないと感じています。MMLでの表現は，入力だけを考えた場合には便利なのですが，素人にはわかりにくいとか，大変そうとか感じてしまいます。今後はなるべくビジュアル化していってほしいです。

森崎　剛(21)　X68000XVI，PC-9801RX21　広島県

●特集に触発されて，生まれて初めて（?）自分で音楽（MML）をやってみました。小・中学校と音楽は3以上でなかった私には大変でしたが，それでも「OPMD」の頃とは比べものにならない表現力の豊かさを実感しました。Z-MUSICシステムのおかげで，敷居がかなり低くなったと思います（といっても高等テクはもてあましぎみ）。

これからZ-MUSICシステムはよりよいものになっていくことでしょう。そこで，これからはMMLからアプローチしていくのではなく，もっと簡単に楽譜から入力できるシステムを作ってほしいと思います。

八亀　桂一(19)　X68000PRO　神奈川県

●「"実戦！"ゲーム作りのKNOW HOW」です。ラスタースクロールについては，いろいろなところで原理が書かれています。そのために新鮮味には欠けましたが，丁寧に説明されていてわかりやすかったです。

ゲームをやっていると，ここの処理はどうやっているのだろうと疑問に思うことが多いので，今後が楽しみです。実際にそのテクニックを自分が使うかどうかは別として，そうしたテクニックを知ることは，精神衛生上よさそうです。個人的には，スプライト関係のものを期待しています。

北風　保(22)　X68000 ACE　東京都

●「"実戦！"ゲーム作りのKNOW HOW」が始まりました。いきなりラスタースクロールということで，思いっきり応用技を披露してくれました。ゲームの作り方というと，キャラクターと背景の重ね合わせやキー入力関係などの基礎を説明して終わりがちです。たまにはこんなテクニック集もよいでしょう。

ただ，ちょっと残念なのは今後の予定がグラフィック中心でサウンド面がおざなりになりそうなことです。PDSのゲームでも音がまったくなかったり，BGMと効果音のどちらか一方だけだったりすることがあります。「ウィザードリィ」のようなRPGやSLGはともかく，STGやACTでは音の有無は死活問題です。画面表示は目に見えるだけあって直観的に理解することもできますが，音のほうはいまひとつわかりません（たとえば，ステレオになっているBGMはどうやっているのだろう）。第2部や番外編という形でもかまいませんからサウンド面のKNOW HOWも教えてください。

中村　健(23)　X68000ACE，PC-386GS，AMIGA500　埼玉県

●「知能機械概論」にあったように，確かに日本語処理はワープロのほうがいいと思います。お世辞にも漢字TALK7は良質のOSとはいえないでしょう。68030の25MHz機のMacintoshを触ったのですが，「重い」のなんの。クイックタイム対応のソフトには「推奨68040」なんて記載されていましたし……。

ハードスペックはそう捨てたものでもないのですが，OSが足を引っ張りすぎのような気がします。そのOSがないと日本語処理できないのですから，Macintoshではしないほうがいいというのは，あながち間違いではないと思います。でもだからといって「じゃあ，Macintoshはクソマシン」と結論をくだすのはそれこそおかど違いというものでしょう。マシンにだって得手不得手はあるものですから。ですよね，有田さん？

中矢　史朗(23)　X68000ACE-HD，X68030　愛媛県

●「EPA2補講（その2）」が興味深い。実は以前に我流ながら「森山風爆発」に挑戦してみたことがある。真っ暗な宇宙空間に球状の爆発が広がり，あたりは閃光に包まれる……はずだった。ところが実際に出来上がったのは，白っぽい円がだんだん大きくなるだけのつまらない映像であった。

そういうわけで，とても期待していたのだが，使うツールや実現までの手順にはそれほど大きな違いはなさそうだ。では，どこが違うのか？　やはり「手描きでゴリゴリ」の部分であろう。コツが少しだけ書かれていたが，私のような凡人にはまだまだ足りない。ツールの使い方などとは違って説明しづらい部分だとは思うが，わかりやすい説明を期待する（なんかエラそう）。

吉岡　洋明(20)　X68000 PRO，PC-8801 MA，FM-NEW7　埼玉県

●「こちらシステムX探偵事務所」が印象に残りました。なぜなら，目的と手段，そして見る側と作る側，話題が先行してしまうということ。それらは，「モーフィング」だけに当てはまるのではなく，私たちをとりまく多くの事象に含まれる問題のような気がしたからです。

橋本　和典(26)　X68000XVI　東京都

ごめんなさいのコーナー

1月号　Z-MUSICシステムver.2.0

P.72　MIDIイベントの簡単な例で，「SC-55のチャンネル1で……」と掲載されています。しかしこれはチャンネル2の場合でした。ただしくは上の例が，

```
00 B0 07 78   ボリューム
00 90 3C 64   ノートオン
60 90 3C 00   ノートオフ
```

となり，下の例が，

```
00 B0 07 78
00 90 3C 64
60 3C 00
```

でした。ご迷惑をおかけしたことをお詫び申し上げます。

2月号　マッドストーカー

P.30　本製品の価格が8,800円となっておりましたが，ただしくは7,800円でした。読者や関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

1月号　(善)のゲームミュージックでバビンチョ

P.85　「ナムコビデオゲームグラフィティVol.10」の発売元は，ビクターエンタテインメントです。関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

バグに関するお問い合わせは
☎03(5642)8182(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

あかりをつけましょ X68000に

▶数少ない女性読者のための付録ディスク「ひなまつりPRO-68K」です(嘘)。今回のディスクはツールが中心ですが，楽しんでみてください。

付録ディスクは5インチ版のみですが，希望者には3.5インチ版への交換サービスを行います。以下のものを同封のうえ，右のあて先(「3.5インチ付録ディスク」係)にお送りください。

1) 130円切手2枚

2) ご自分の住所・氏名を書いた宛名シール

また，発送には多少時間がかかりますが，あらかじめご了承ください。

▶さて，来月号で予定されている「1993年度GAME OF THE YEAR」。今回はどんなゲームがOh!Xゲーム大賞に選ばれるのでしょうか。非常に楽しみです。最近，元気のないX68000ゲーム業界ですが，そんな辛気臭い雰囲気を吹き飛ばすくらい勢いのある「GAME OF THE YEAR」にしたいですね。

なお，投票方法に不明確なところがあったことをお詫びいたします。

▶5月号では読者参加の「言わせてくれなくちゃだワ」を今年も行う予定です。ということで，いつものとおり今月号にはこの「ちゃだワ」用のアンケート用紙が綴じ込まれています。これは，読者の皆さんが，どのような意識でもってコンピュータ，そしてOh!Xとつきあっているかを知る資料となります。また，アンケートハガキに書ききれない意見を発表する絶好の機会です。主役は読者の皆さん，というこの企画にぜひ，ご協力ください。

▶本誌1993年10月号の「SLASH Ver.1.0」および11月号の「ポリゴン描画のためのエッジ検出法」の記事中において，「ポリゴナイザ」という言葉を一般名称のように使用していますが，「ポリゴナイザー／POLYGONIZER」は株式会社ナムコが商標権をもつ登録商標です。読者に誤解を与えるとともに，関係者各位にご迷惑をおかけしましたことをお詫びいたします。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103 東京都中央区日本橋浜町3-42-3
ソフトバンク出版部
Oh!X「テーマ名」係

SHIFT・BREAK

▶年の瀬にわざわざ晴海まで足をお運びくださった皆様，どうもありがとうございました。おかげで上昇気流5は無事完売いたしました！　感謝感謝。なんか一度で全部売れちゃったのは初めてだったので，驚いたりしてるんですが……ということで，あと上昇5が手に入るのは今月のプレゼントしかありません。皆様どうぞふるってご応募ください(笑)。(哲)

▶以前からACEのノロさに我慢がならなかったので思い切ってCompactXVIを買った。家賃滞納覚悟でHDDも買って「あとはメモリを装着するだけでカンペキだよーん！　うっしっし(これも死語か？)」と鼻息荒くバイクで高速をかっとばして秋葉原まで行ったら「XVIのメモリ品切れ」という札が。ええっ，1カ月待ち？　ガク……もういらないやい。　(H)

▶環境にやさしいパソコンが出てきた。消費電力を抑えるのが環境にいいらしい。部屋の暖房がパソコンなんて話を聞くから，かなりの節電にはなるんだろうけど，半年で時代遅れになるようなモノが「環境にやさしい」といってもねぇ。本体が食べられるんなら別だけどさ。製品寿命の長い製品を選んだほうがよっぽどエコロジーなんじゃない？　(E.K.)

▶北海道の伊藤博則さんほか何名かの方から，12月号で書いた悪夢を診断したお手紙をいただきました。ありがとうございます。いわれてみるとなんだか思い当ることが多い……ちょっと驚きです。でも最近は寝ると完全に熟睡してしまって夢を見ることもなかったりしていいような悪いような……。意図的にいい夢見る方法ってないもんですかねぇ？　(で)

▶1月末に沖縄に遊びに行った。現地の友人に，彼女のお気に入りの場所を案内してもらったのでおいしいところを満喫できた。冬でも暖かいし海は綺麗だし，住み着きたいくらい気に入ってしまったのだ。遅ればせながらWORLD CIRCUITの鈴鹿タイムアタックにはまっている。とりあえず1分34秒台には突入したが……。(渡り鳥生活にも憧れているA.T.)

▶Jリーグの名場面集はいいけれど，プロ野球の激突や乱闘シーンを楽しむのと同じノリでサッカーのそれを楽しもうというのはあまりに失礼だ。もともと激突しないことが前提の野球と，しじゅう身体を触れ合わせてプレーするサッカーを同列にしてどうするのだ？　プロ野球番組と同じ構成でサッカー番組を作って視聴率稼げてラッキーとは，とほほ。(K)

▶先々月より，なかなか体調が戻らないので不安神経症との診断に疑いをもつ。今度は脳神経外科に行く。とりあえず頭部のCTを撮影したが異常なし。次は脳波の検査。測定中に電極のチェックを何度もしていると思ったら案の定，波形がおかしいという。CTの解像度は低いので今度はMRIで測定することになった。さて，その結果は。　(以下次号のKO)

▶時刻はAM4:30。成田空港では雪が降っているらしい。友人に会社から拾ってもらい一路茨城へ。ふと気がつくと外が白い。一歩車から足を踏み出すと，寒さに身が凍る。ここは美浦トレーニングセンター。馬の調教見学にやってきた。厩舎を訪問して話を聞いたが，ここでの生活は私には無理だと感じた。だって朝が……。(今度は栗東にも見学にいきたい高)

▶7年ぶりに生活のなかに自転車が導入された。バイクも車も運転できない「優良ドライバー」のあたしにとって，唯一「乗れるもん」と胸張っていえる乗り物だ。これで遠くのお買い物も大きくて重い荷物も夜のお散歩もへっちゃらさ。さあ次なる楽しみはアコガレの自転車通勤。はやく春にならないかなぁ。でも名前はまだ，ない。　(ふ)

▶SION4(仮)は順調に遅れています。大口たたいたくせにだらしないのですが，時間が取れないなどと泣き言いって逃げ回っています。ま，それはさておき，優秀なモデラがほしい今日この頃。誰か，制作していません？　もしも，制作している人がいたらOh!Xまでご一報を。なお，「SLASHver.2.0」は5月号に掲載が延期されました。　(J)

▶正月に帰省すると，もう帰ってくるなといわれた。世間では勘当されたというのだろうか？　で，MPEG4は9600bpsくらいの通信速度でフルモーションの画像と音声を再現する技術になるという。いまの1/1000のビットレートだが，その根拠が「1ビットあたりの処理に1000倍時間がかけられるから」というのはなんか変だ。　(U)

▶例のビデオCDだけど，アメリカだとCD-Iのソフトとして映画のタイトルなんかも結構売られている。映画だと2枚組(1枚74分)で25ドルというから日本では音楽CD1枚分の値段だ。安いのも魅力だが，ディスク2枚が1枚分のCDケースに入っているのには驚いてしまった。これは素晴らしい。レーザーディスクだと重いし場所とるんだよね。　(T)

microOdyssey

自転車の名前は目下検討中だが，パソコンにはすでにみんな名前がある。子供の頃うちにあった人形も全部名前を持っていた。いつのまにか違う名になっていることもあるくらいで名前そのものにはたいした意味はないのだが，命名することには意味がある。ひとつの愛情の表れでもある。そのものに対するいとおしさだったり，それをくれたひとへの感謝だったりする。

で，話題の悪魔ちゃんである。冗談だと思ったら，ご両親は本気だったのね，とちょっと驚いた。何に違反しているわけでも誰に迷惑をかけたわけでもないから，法務省が介入云々はそれはそれで理不尽な話ではある。だけど，やあほんとにつけちゃいましたね，ってのが感想。

しかし，他人の子供の名前に議論百出，ってのもいかにも平和な日本らしい。週刊誌やら新聞やらにもいろいろな珍名，奇名が紹介されて，昔はもっといい加減な名前がたくさんあった，なんて話も聞いた。要は本人が気にいるかどうかのようだが，「名前に負けないように」とか「出会いが増える」という主張もなんだかおかしい。名前って，そんなものなのだろうか。

お伽ばなしのなかの小人は，名前を言い当てられて魔力を失い，赤ん坊を奪い取ることができなかったし，倭武尊命も自ら名乗ったために命を落としたのだった。「キャッツ」にも「猫の本当の名前は誰も知らない」という一節が出てくる。名前には魂が宿るという思想はあちこちにあり，唱えることで力を得たり，名前を知られることで力を失ってしまったりするらしい。「体を表す」というくらいで，「たかが名前」ではないようだ。だから誰だって自分の名前には愛着を持つし，命名には願いを込める。

しかし，魔力云々は別にして現実問題としては，名前というのは本人のものではあるが，ある意味では自分だけのものでもない。識別番号の一種，とは言わないまでも，半分くらいは本人以外の人が使うことになる。そう考えると，いくら他人の名前でも「アクマさん」とはちょっと呼びにくい。目上でこだわりのない人ならいいが，こちらが配慮すべき立場だった場合はやはり戸惑ってしまうだろう。一対一のつきあいならまだいいが，集団のなかだったら，困ることもあるような気がする。第一，本当にその人をののしりたいときはどうすればいいのか。「この悪魔！」ったたってねぇ……。

さて，悪魔ちゃんだが，命名といえば必ず問題になるはずの「画数」についてはどうなのだろうか。少なくとも私はそれについての記事は見なかったのだが，興味がある。お父さんがひとこと「これは画数が最高なので決めました」といったら，反対者の何人かは口をつぐんでしまうのだろうか。少なくとも「子供のためを思って」という「愛情の証明」のひとつにはなるのかもしれない。それにしても，もしも「悪魔」とか「幽霊」とかが縁起のよい画数だったとしたら，なんだか楽しい。

ところで，私だったら「悪魔と名づける」なんて言われたら即離婚する！　といいきまいていたら「確かめてから結婚するの？」との質問。

う～ん，いままでは考えもしなかったけれど，前例もできたことだし，やっぱり結婚の条件には加えておくべきか。「三高」なんてのより，よっぽど重大な問題だと思うけどね。　（ふ）

1994年4月号3月18日（金）発売

特集　X68000の仲間たち

・X68000で使用できる周辺機器のレポートなど

1993年度 GAME OF THE YEAR

新製品紹介

・SX-WINDOW開発キットWorkroomSX-68K

・SX-WINDOW用スケジュール管理ソフトDoubleBookin'

第6回アマチュアCGAコンテスト結果発表

バックナンバー常備店

地域		店名	電話
東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートBI	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3209)0656
	渋谷	大盛堂書店	03(3463)0511
	池袋	旭屋書店池袋店	03(3986)0311
	八王子	くまざわ書店八王子本店	0426(25)1201
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X　3月号

■1994年3月1日発行　特別定価800円(本体777円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部 ☎03(5642)8122
販売局 ☎03(5642)8100 FAX 03(5641)3424
広告局 ☎03(5642)8111
■印　刷　凸版印刷株式会社

バックナンバー案内

BACK ISSUES

ここには1993年3月号から1994年2月号までをご紹介しました。現在1992年6，7，12，1993年6～12，1994年1～2月号の在庫がございます。バックナンバーはお近くの書店にご注文ください。定期購読の申し込み方法は148ページを参照してください。

1993

3月号（品切れ）
特集　X-BASICを学ぶ
連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ANOTHER CG WORLD/ハード工作
ショートプロ/Computer Music入門/Z80's Bar
●緊急速報　32ビットマシンX68030
●新製品紹介　音源モジュールSC-33/GS音源搭載JW-50
LIVE in '93　ストリートファイターII/晴れたらいいね 他
THE SOFTOUCH　究極タイガー/チェルノブ/シムアント 他
全機種共通システム　シューティングゲームコアシステム作成法(1)

4月号（品切れ）
特集　X68第7世代へ
連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
ハード工作/吾輩はX68000である/Computer Music入門
●決定！　1992年GAME OF THE YEAR
●名作ゲーム再遊記
LIVE in '93　FIGHTMAN/ミンキーモモより 愛しのマーシカ
THE SOFTOUCH　スターフォース/元朝秘史　他
全機種共通システム　シューティングゲームコアシステム作成法(2)

5月号（品切れ）
特集　襲撃！　SX-WINDOW
第8回　言わせてくれなくちゃだワ
連載　DōGA CGアニメーション講座/ANOTHER CG WORLD
響子 in CGわ～るど/ショートプロ/大人のためのX68000
ハード工作/吾輩はX68000である/Computer Music入門
●X68030へのソフトウェア対応について
LIVE in '93　MAGICAL SOUND SHOWER/もう笑うしかない　他
THE SOFTOUCH　エトワールプリンセス/メガロマニア　他
全機種共通システム　シューティングゲームコアシステム作成法(3)

6月号
創刊11周年特別企画　確率遊技シミュレーション
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/大人のためのX68000
ハード工作/吾輩はX68000である/Computer Music入門
●新製品紹介　SC-55mkII
LIVE in '93　ストリートファイターIIより 春麗のテーマ/
BAY YARD/LOVE&CHAIN
THE SOFTOUCH　餓狼伝説/信長の野望・覇王伝　他
全機種共通システム　REVERSI

7月号
特集　席巻するローテク文明
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/マシン語プログラミング
ハード工作/吾輩はX68000である/Computer Music入門
●新製品紹介　ドローイングパッド33070&MATIER
LIVE in '93　Midnight Circle/今日の日はさようなら/赤い靴
THE SOFTOUCH　悪魔城ドラキュラ/リブルラブル/大航海時代II/
銀河英雄伝説III/幻影都市/ヴェルスナーグ戦乱
全機種共通システム　MSX用S-OS "SWORD"

8月号
特集　C言語実践的入門
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/Computer Music入門/大人のためのX68000
吾輩はX68000である/ショートプロ/ANOTHER CG WORLD
●特別企画　夏真っ盛り，アマチュアリズムのX68000
LIVE in '93　SPLASH WAVE
THE SOFTOUCH　悪魔城ドラキュラ/リブルラブル/餓狼伝説/
ロボットコンストラクションR.C./Winning Post
全機種共通システム　MACINTO-C再掲載

9月号
特集　光学式磁気円盤MO
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/大人のためのX68000
ハード工作/Computer Music入門/ANOTHER CG WORLD
●新製品紹介　OS-9/X68030
LIVE in '93　ファイナルファンタジーVのテーマ/銀河鉄道999/
アルスラーン戦記IIより　汗血公路/ちょうちょ
THE SOFTOUCH　悪魔城ドラキュラ/コットン/ダーク・オデッセイ 他
全機種共通システム　7並べ/SLANG再々掲載

10月号
特別企画　秋祭りPRO-68K
連載　ハードコア3D/Computer Music入門/マシン語プログラミング
DōGA CGアニメーション講座/こちらシステムX探偵事務所
響子 in CGわ～るど/ショートプロ/吾輩はX68000である
●特別付録　秋祭りPRO-68K（5"2HD）
●SCSIパックンTOWER JACK
LIVE in '93　未来予想図II/OutRunより　PASSING BREEZE
THE SOFTOUCH　コットン/The World of X68000/あにまーじゃんV3
全機種共通システム　シューティングゲームコアシステム作成法(4)

11月号
特集　ポリゴナイザSLASHの活用
連載　ハードコア3D/Computer Music入門/ファイル共有の実験と実践
こちらシステムX探偵事務所/目指せジョイスティックの星
響子 in CGわ～るど/ショートプロ/大人のためのX68000
●新製品紹介　Easydraw SX-68K
OS-9 Ultra C/Technical Tool Kit
LIVE in '93　渚のアデリーヌ/エロティカ・セブン
THE SOFTOUCH　ぷたさん/ダイアット・ヴァークス
全機種共通システム　S-OSで学ぶZ80マシン語講座(1)

12月号
特集　古今東西ゲーム議論
連載　ハードコア3D/マシン語プログラミング/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ファイル共有の実験と実践
●新製品紹介　MATIER ver.2.0
C Compiler PRO-68K ver.2.1 NEW KIT
LIVE in '93　クリスマス・イブ/星に願いを
THE SOFTOUCH　ネメシス'90改/項劉記/スーパーリアル麻雀PII＆PIII
全機種共通システム　エディタアセンブラREDA再掲載

1994

1月号
特集　Z-MUSICシステムver.2.0
連載　ハードコア3D/ゲーム作りのKNOW HOW/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ファイル共有の実験と実践
●特別企画　ANOTHER CG WORLD in Hong Kong
LIVE in '94　LAST WAVE/スターウォーズ/明日への扉/夢路より　他
THE SOFTOUCH　ストリートファイターIIダッシュ/餓狼伝説2/
ドラゴンバスター/X68000傑作ゲーム選
全機種共通システム　S-OSで学ぶZ80マシン語講座(2)

2月号
特集　X-BASICとグラフィック
連載　ハードコア3D/ワンチップIC/響子 in CGわ～るど
DōGA CGアニメーション講座/こちらシステムX探偵事務所
ショートプロ/Computer Music入門/ANOTHER CG WORLD
●新製品紹介　ハイパーピクセルワークス
LIVE in '94　ランス3/新宿駅，巣鴨駅の発車メロディ/ピコー・ソング
THE SOFTOUCH　キーパー/マッドストーカー/餓狼伝説2　他
全機種共通システム　S-OSで学ぶZ80マシン語講座(3)
YGCSver.0.20リファレンスマニュアル

満開の電子ちゃん

作・え 岡村祭

69号は既刊（94年1月18日発売）です。この号からの定期購読もできます。その旨をお書き添えください。

桂 尚恵（茨城県）

X68kを持っていて、電脳倶楽部を知らない人はいないと思うのですが、定期購読していない人ってどのくらいいるのでしょう？ 私はプログラムも組めないし、絵も描けない、音楽もできないです。でも、そんな私でも電クラに参加できるのです。そうして、私のX68kはパワーアップしてゆき、私もいろいろ判るようになりました。

私が電クラに何をしているっていう訳ではないけれど、参加している充実感があって、ああ、定期購読していてよかったって毎月18日に思うんです。

マイコン専門ショップ

P&A

SHARPエキスパートショップ

今が購入のチャンス！

SHARP

2/18〜3/17

注目!!夏のボーナス一括払い手数料（金利）無料（平成6年3月末／4月末／5月末／6月末／7月末のいずれかをご指定ください。）

X68000 Compact XVI

旧シリーズ今が買いどき!!
（クレジット表：送料・消費税込み）送料¥2,000・消費税別

① 本体＋モニター

●CZ-674C-H
●CZ-608D-H
定価¥392,800
P&A超特価 ¥158,000

12回	14,500	24回	7,700	36回	5,300	48回	4,200	60回	3,500

② 本体＋モニター＋FDD(5″×2)

●CZ-674C-H
●CZ-608D-H
●CZ-6FD5(FDD)
定価¥492,600
P&A超特価 ¥203,000

12回	18,500	24回	9,800	36回	6,800	48回	5,300	60回	4,500

③ 本体＋モニター（TVチューナー付）

●CZ-674C-H
●CZ-614D-TN
●CZ-6CR1(RGBケーブル)
●CZ-6CT1(TVコントロール)
定価¥443,000
P&A超特価 ¥199,000

12回	18,200	24回	9,600	36回	6,700	48回	5,200	60回	4,400

④ 本体＋モニター（TVチューナー付）＋FDD(5″×2)

●CZ-674C-H
●CZ-614D-TN
●CZ-6CR1(RGBケーブル)
●CZ-6CT1(TVコントロール)
●CZ-6FD5(FDD)
定価¥542,800
P&A超特価 ¥247,000

12回	22,500	24回	11,900	36回	8,300	48回	6,500	60回	5,400

X68000 Compact XVI

本体（単品）

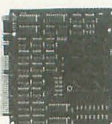
⊙CZ-674C
定価¥298,000
P&A超特価 ¥98,000

X68000 PRO II

本体（単品）

⊙CZ-653C-BK
定価¥285,000
P&A超特価 ¥68,000

■モニター変更の場合 ※Compact XVI①・②のモニターを、
●CZ-607D-TN（定価¥ 99,800）に変更の場合¥ 3,000加算して下さい。
●CZ-621D(B)··（定価¥168,000）に変更の場合¥58,000加算して下さい。

X68030/68000メモリボード(I/Oデータ)

①SH-5BE4-8M(X68030用)…………（送料・消費税込み¥47,586）特価¥45,500
②SH-6BE1-1ME(600C専用)…………（送料・消費税込み¥11,845）特価¥10,800
③1MB増設RAMボード(ACE/PRO/PROII用)（送料・消費税込み¥11,845）特価¥10,800
④2MB増設RAMボード(拡張スロット用)·（送料・消費税込み¥24,205）特価¥22,800
⑤4MB増設RAMボード(拡張スロット用)·（送料・消費税込み¥40,170）特価¥38,300

モデム

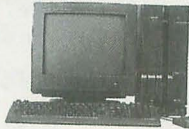
（送料¥1,000）

マイクロコア ●MC-14400FX…………（定価¥46,800）▶特価¥34,500
富士通 ●FMMD-3111G…………（定価¥35,800）▶特価¥24,800
オムロン ●MD-24XT10V…………（定価¥29,800）▶特価¥22,500
●MD-96XT10V…………（定価¥46,800）▶特価¥32,000
アイワ ●PV-AF144V5…………（定価¥64,800）▶特価¥49,000

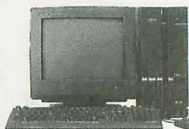
●本広告の掲載の商品の価格については、消費税は含まれておりません。

X68030お買い得セット

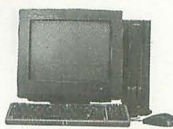
（クレジット表：送料・消費税込み）

①X68030

●CZ-500C ●CZ-608D
定価合計¥492,800
P&A超特価
¥304,000

12回	27,800	24回	14,700	36回	10,200	48回	8,000	60回	6,700

②X68030 HD

●CZ-510C ●CZ-608D
定価合計¥582,800
P&A超特価
¥403,000

12回	36,800	24回	19,400	36回	13,500	48回	10,500	60回	8,900

③X68030 Compact

●CZ-300C ●CZ-608D
定価合計¥482,800
P&A超特価
¥333,000

12回	30,400	24回	16,100	36回	11,100	48回	8,700	60回	7,300

④X68030 Compact HD

●CZ-310C ●CZ-608D
定価合計¥572,800
P&A超特価
¥398,000

12回	36,300	24回	19,200	36回	13,300	48回	10,400	60回	8,700

■モニターの変更
（※300シリーズにチューナー付のモニターを接続の場合CRTケーブルを購入して下さい。）
①CZ-607D(チューナー付)
……に変更の場合¥ 3,000加算して下さい。
②CZ-614D(チューナー付)
……に変更の場合¥31,000加算して下さい。
③CZ-621D(B)
……に変更の場合¥58,000加算して下さい。

X68030 発売記念

X68030をモニターとセットまたは単品で購入の方さらに現在お持ちのパソコンと、下取り交換されたお客様に期間中もれなく、
①サイバースティック (CZ-8NJ2 ¥23,800)
②X-68000 フロッピーアタッシュケース（¥8,000）とクリスタルボルシェ（¥8,000）
以上のいずれかプレゼント.!!

68000パワーアップキット（ジャスト）

HARP＋ER10Sでメモリーアクセス 約33% UP

⊙MPUアクセラレータ H.A.R.P…………特価¥25,000
（600C、ACE、EXPERT、PRO、SUPER用）
⊙I/O拡張スロット ESX68L4…………特価¥33,500
（2スロットに拡張、全機種対応）
⊙拡張SIMMメモリーボード ER10S…………特価¥12,500
（SIMM未実装タイプ、SIMMソケット×2 全機種対応）
●増設SIMM ●HT04MB-70-DV(加賀電子)…………特価¥18,200
●HT08MB-70-DV(〃)…………特価¥35,700
(例1)X68000に8M増設 ER10S＋HT08MB-70-DV＝¥48,200(DOS V用 72ピン、70ns)
(例2)最大メモリ実装(12M) ER10S＋HT08MB-70-DV＋HT04MB-70-DV＝¥66,400

X68000/68030専用ハードディスク

（送料¥1,000・消費税別）

外付

■富士通
⊙FMHD-1201G(120MB、17ms)……定価¥ 70,000▶特価¥49,800
⊙HD-K240(モッキンバード)(240MB、15ms)……定価¥ 79,800▶特価¥49,800
⊙HD-K540(モッキンバード)(540MB、10ms)……定価¥148,000▶特価¥98,000

■ロジテック
⊙SHD-FMX240(240MB)(ケーブル付)
…………定価¥138,000▶特価¥57,800

■ジェフ
⊙GF-240e(240MB、15ms、64K)……定価¥118,000▶特価¥49,800
⊙GF-340i(340MB、14ms、64K)……定価¥158,000▶特価¥59,800
⊙GF-540i(540MB、8.5ms、256K)…定価¥238,000▶特価¥108,000

■Filo(ファイロ)
⊙CS-H300(330MB、12ms)…………特価¥59,800
⊙CS-H500(520MB、12ms)…………特価¥89,800

内蔵

■CZ-500C/300C専用
⊙CZ-5H08(80MB/23ms)
…………定価¥ 98,000▶特価¥71,800
⊙CZ-5H16(160MB/18ms)
…………定価¥135,000▶特価¥99,500

ズバリご奉仕

P&Aならではの 新品パソコン5年保証

《業界No.1の"P&Aメンテナンスサポート"》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証!! ※一部商品は除きます。）
②中古パソコンの1年間保証（※モニター・プリンター6ヶ月間保証!!）
③初期不良交換期間3ヶ月（※新品商品に限らせていただきます。）
④永久買取保証
⑤配達日の指定OK!!（土曜・日曜・祭日もOK!!）
⑥夜間配達もOK!!（※PM6:00〜PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利!!
③月々の支払いは￥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括支払いOK!!
⑩商品到着払いOK!!（代引き手数料が必要になります。10万円まで900円）
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

●法人向け リースシステム
業務に最適なシステムを構築します。
損金処理が可能なリース契約をどうぞ。

周辺機器コーナー （送料￥1,000・消費税別）

カラーイメージスキャナ

■JX-220X《限定》
定価￥168,000
特価￥89,800

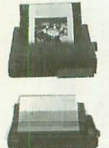
■JX-325X
定価￥190,000
特価￥143,000

カラーイメージジェット

■IO-735X-B
定価￥248,000
特価￥128,000

FDD(5インチ×2基)

■CZ-6FD5
定価￥99,800
P&A超特価
￥49,800

漢字プリンター（ケーブル用紙付）

■CZ-8PC5-BK
定価￥96,800
▶**特価￥38,000**

■CZ-8PK10
定価￥97,800
▶**特価￥71,000**

光磁気ディスク（X68000用）

■CS-M120（コバル）
●ケーブル、ターミネータ付 ￥178,000
特価￥119,000

■LMO-FMX330
●ケーブル、ターミネータ付 ￥178,000
特価￥135,000

●CZ-6TU………定価￥ 33,100▶特価￥ 23,900
●CZ-8NM3……定価￥ 9,800▶特価￥ 7,200
●CZ-8NT1……定価￥ 13,800▶特価￥ 10,000
●CZ-6BE2A……定価￥ 59,800▶特価￥ 42,800
●CZ-6BE2B……定価￥ 54,800▶特価￥ 39,300
●CZ-6BE2D……定価￥ 54,800▶特価￥ 39,300
●SH-6BF1……定価￥ 49,800▶特価￥ 36,500
●CZ-6BP1……定価￥ 79,800▶特価￥ 57,000
●CZ-6BM1……定価￥ 26,800▶特価￥ 19,300
●CZ-6SD1……定価￥ 44,800▶特価￥ 32,500
●SH-6BN1……定価￥ 29,800▶特価￥ 21,800
●CZ-6BV1……定価￥ 21,000▶特価￥ 15,200
●CZ-6BC1……定価￥ 79,800▶特価￥ 57,000
●SH-6BG1……定価￥ 59,800▶特価￥ 43,800
●CZ-6PV1……定価￥198,000▶特価￥142,000
●CZ-6BS1……定価￥ 29,800▶特価￥ 21,500
●CZ-8NJ2……定価￥ 23,800▶特価￥ 17,500
●CZ-6BL2……定価￥298,000▶特価￥214,000
●JX-220X……定価￥168,000▶特価￥ 89,800
●CZ-6CS1（674C用）定価￥ 12,000▶特価￥ 8,900
●CZ-68HA……▶特価￥ 91,000
●CZ-6CR1（RGBケーブル）・定価￥ 4,500▶特価￥ 3,600
●CZ6CT1（テレビコントロール）・定価￥ 5,500▶特価￥ 4,400
●CZ-6BP2……定価￥ 45,800▶特価￥ 33,300
●CZ-5MP1（X68030用）・定価￥ 54,800▶特価￥ 42,000

■システムサコム ボード
●SX-68MII（MIDI）
定価￥19,800▶特価￥13,500
●SX-68SC(SCSI)
定価￥26,800▶特価￥17,500

（X68030用）
●CZ-5BE4
定価￥54,800 ▶￥42,000
●CZ-5ME4
定価￥49,800 ▶￥38,000

X68000用ソフトコーナー （送料￥700・消費税別）

●Z's STAFF PRO68K Ver.3.0（ツァイト）……定価￥58,000▶特価￥37,500
●Z's TRIPHONYデジタルクラフト（ツァイト）……定価￥39,800▶特価￥27,000
●テラッツォ（ハミングバード）……定価￥19,400▶特価￥13,600
●ラジックパレット（ミュージカルプラン）……定価￥19,800▶特価￥14,200
●たーみのる2(SPS)……定価￥17,800▶特価￥13,000
●Mu-1 Super（サンワード）……定価￥39,800▶特価￥28,500
●CMA68K（シティソフト）……定価￥29,800▶特価￥21,800
●サイクロンEXPRESS α68……定価￥98,000▶特価￥69,000
●C-TRACE68 Ver.3.0（キャスト）……定価￥98,000▶特価￥68,500
●OS-9/X68030 V.2.4.5（マイクロウェアシステムズ）……定価￥25,000▶特価￥19,900
●C&Professional Pack V.3.2（マイクロウェアジャパン）……定価￥80,000▶特価￥57,800
●ウエットペイント1〜3（ウエーブトレイン）（各）……定価￥15,000▶特価￥11,500
●マチエール Ver.2.0……定価￥39,800▶特価￥28,800
●Windex PRO68（JEL）……定価￥28,000▶特価￥20,500
●CZ-213MSD MUSIC PRO68K……定価￥18,800▶特価￥13,200
●CZ-214MSD SOUND PRO68K……定価￥15,800▶特価￥11,300
●CZ-215MSD Sampling PRO68K……定価￥17,800▶特価￥12,500
●CZ-220BSD DATA PRO68K……定価￥58,000▶特価￥40,000
●CZ-225BSV Multiword Ver.2.0……定価￥32,000▶特価￥23,000
●CZ-243BSD CYBERNOTE PRO68K……定価￥19,800▶特価￥15,000
●CZ-247MSD MUSIC PRO68K(MIDI)……定価￥28,800▶特価￥20,500
●CZ-249GSD CANVAS PRO68K……定価￥29,800▶特価￥22,000
●CZ-251BSD Hyperword……定価￥39,800▶特価￥29,400
●CZ-253BSD CARD PRO68K Ver.2.0……定価￥29,800▶特価￥22,700
●CZ-257CSD Communication PRO68K Ver.2.0……定価￥19,800▶特価￥15,300
●CZ-258BSD Teleportion PRO68K……定価￥22,800▶特価￥16,900
●CZ-261MSD MUSICstudio PRO68K Ver.2.0……定価￥28,800▶特価￥21,200
●CZ-263GWD Easypaint SX-68K……定価￥12,800▶特価￥ 9,800
●CZ-264GWD Easydraw SX-68K……定価￥19,800▶特価￥15,300
●CZ-265HSD NewPrint Shop Ver.2.0……定価￥20,000▶特価￥15,400
●CZ-266BSD PressConductor PRO68K……定価￥28,800▶特価￥22,000
●CZ-267BSD CHART PRO68K……定価￥38,000▶特価￥29,800
●CZ-272CWD Communication SX68K……定価￥19,800▶特価￥14,500
●CZ-275MWD SOUND SX68K……定価￥15,800▶特価￥11,500
●CZ-284SSD OS-9/X68000 Ver.2.4……定価￥35,800▶特価￥25,600
●CZ-286BSD BUSINESS PRO68K……定価￥28,000▶特価￥20,500
●CZ-288LWD開発キット（workroom）……定価￥39,800▶特価￥29,700
●CZ-290TWD SX-WINDOW ディスクアクセサリー集……定価￥14,800▶特価￥11,500
●CZ-294SS(5")/SSC(3.5") SX-WINDOW Ver.3.0……定価￥19,800▶特価￥15,200
●CZ-295LSD C-Compiler PRO68K Ver.2.1 NEW KIT……定価￥44,800▶特価￥32,500

☆ゲームソフト25%OFF OK!!（一部ソフト除く）

全国通販

★頭金なし!
★即日発送

●お近くの方はお立寄り下さい。専門係員が説明いたします。
●本体単品で特価で受付します。詳しくは電話にてお問合せ下さい。
●ビジネスソフト定価の20%引きOK!TELください。

P&A特選 今月の中古特選品

●CZ-600C‥￥55,000
●CZ-601C‥￥65,000
●CZ-611C‥￥70,000
●CZ-652C‥￥75,000
●CZ-612C‥￥95,000
●CZ-603C‥￥85,000
●CZ-653C‥￥78,000
●CZ-612C‥￥90,000
●CZ-623C‥￥110,000
●CZ-674C‥￥108,000
●CZ-634C‥￥130,000
●CZ-644C‥￥178,000
※上記は単品価格、モニター別売。

新古品 限定
●CZ-674CH
●CZ-608DH
￥138,000

中古品
●CZ-674CH
●68000専用モニター付
￥128,000

限定
●CZ-634CTN（チタン）（中古）
●CZ-613D（グレー）（新品）
￥190,000
（モニターをCZ-614TN（チタン）に変更の場合￥20,000加算）

中古品
●CZ-634CTN
●68000専用モニター付
￥158,000

新古品 限定
●CZ-644CTN
●CZ-604DB
￥228,000

中古品
●CZ-644CTN
●68000専用モニター付
￥198,000

中古・高価現金買取り／下取りOK!!

■まずはお電話下さい。
下取り専用 買取り電話▶ ☎03-3651-1884 FAX.03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送りください。

買取り価格…完動品・箱／マニュアル／付属品の価格です。

●下取りの場合…価格は常に変動していますので査定額を電話で確認してください。（差額は、P&A超低金利クレジットをご利用ください。）
●買取りの場合…現品が着き次第、2日以内に高価買取金額を連絡し、振込み、又は書留でお送り致します。
●近郊の方はP&A本店に直接お持ちください。即金にて￥1,000,000までお支払い致します。

●最新の在庫情報・価格はお電話にてお問い合せください。
●買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せください。
●価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認ください。
●本商品の掲載の商品の価格については、消費税は、含まれておりません。
●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せください。

P&A特選パソコンラック&OAチェアー （消費税込み）（送料無料、離島を除く）

①3段 ￥8,240
②4段 ￥9,785
③5段 ￥11,845
(W-640)

※全機種→キャスター付
※上から2番目棚板移動可能（4/5段）
※フレーム色：4段→黒、3/5段→ホワイト

①￥9,270
●布張り（ダークグレー）
●ガスシリンダー

②￥11,330
●布張り（ダークグレー）
●ガスシリンダー
●肘付

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。●1回〜84回払いまで出来ます。但し、1回のお支払い額は￥1,000円以上。

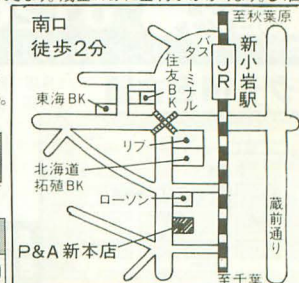
〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。（電信扱いでお振込み下さい。）

〔振込先〕さくら銀行 新小岩支店
当座預金 2408626 (株)ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.9	3.9	4.9	5.4	8.4	11.4	15.9	20.9	26.9	34.9

南口 徒歩2分
至秋葉原 バスターミナル 住友BK JR 新小岩駅 東海BK リブ 北海道拓殖BK ローソン 蔵前通り P&A新本店 至千葉

P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
●営業時間：AM10：00〜PM7：00 日・祭：AM10：00〜PM6：00
☎03-3651-0148（代）
FAX.03-3651-0141
●定休日／毎週水曜日

※お支払いは、便利な商品到着払い〈手数料(10万円まで900円)要〉をご利用下さい。

●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上お申し込み下さい。詳しくは、お電話でお問い合せ下さい。

業界No.1!!

12回払い、7.5%がナント6%に!

クレジット金利がこんなにお安くなりました!!～月々ムリのないお支払い額で欲しかったパソコンがお手元に!!～

支払回数(回)	1	3	6	10	12	15	18	20	24	30	36	42	48	54	60
分割払い手数料率(%)	2.5	3.5	4.5	5.5	6	9	11	12	12.5	16.5	17.5	22	23	28.5	29.5

ツクモグローバルJCBカード登場!!

好評 入会者受付中!　18才以上なら学生でもOK!

JCBならではの国内、海外サービスにツクモオリジナルの特典をプラス。お支払いはプランに合わせて、1回・2回・ボーナス一括・リボルビング払いから選べるのでとても便利!!ツクモ各店備え付けの入会申込書にてお申し込み下さい。詳しくはグローバル事務局03-3251-9898または各店へ。

★ジャックス・VISA・セントラル・マスターも取り扱っております。

超低金利!夏・冬ボーナス二回払い受付中!詳しくは各店までお問い合わせ下さい。

〈大容量記憶装置〉

※SCSIボードが必要な方にはセット価格に¥24,000加算となります。

MO特選セット

Panasonic
- LF-3100B　¥178,000
- MOメディア　本体同梱
- SCSIケーブル　サービス
- **ツクモ特価¥99,800**

SONY
- RMO-S360　¥169,000
- MOメディア　本体同梱
- SCSIケーブル　サービス
- **ツクモ特価 ¥128,000**

富士通OA(Filo)
- CS-M120PX(ブラック)　¥178,000
- SCSIケーブル　サービス
- MOメディア　サービス
- ターミネータ　サービス
- **ツクモ特価¥125,000**

すごいぞCD-ROM!!

CD-ROMドライブ(2倍速)

ELECOM	ECD-250(TOSHIBAドライブ)	ツクモ特価¥47,800
Logitec	LCD-500(SONYドライブ)	ツクモ特価¥56,800
SONY	CDU-7811(SONYドライブ)	ツクモ特価¥49,800
Panasonic	LK-RC533NZ5(松下ドライブ)	ツクモ特価¥49,800

ECD-250

6連装CD-ROMドライブ

PIONEER
- DRM-602X(2倍速)　ツクモ特価¥ 78,000
- DRM-604X(4倍速)　ツクモ特価¥178,000
- CD-ROMドライバーソフト+SCSIケーブル　ツクモ特価¥ 9,200

ハードディスク

VIP-120CX(120MBハードディスク)	ツクモ特価¥ 39,800
VIP-240CX(240MBハードディスク)	ツクモ特価¥ 49,800
VIP-350CX(340MBハードディスク)	ツクモ特価¥ 69,800
540MBハードディスク	ツクモ特価¥105,000～

パソコン通信　※価格はすべてツクモ特価

モデム

AIWA	PV-AF144V5	¥45,800
OMRON	MD144XT10V	¥39,800
MicroCORE	MC1440FX	¥39,800
Panasonic	TO-703B	¥45,800

通信ソフト
- た～みのる2　¥13,000
- Communication SX-68K　¥16,800

ディスプレイも特別価格にて提供中!

CZ-607D(14型カラーディスプレイテレビ)	ツクモ特価¥ 60,000
CZ-608D(14型カラーディスプレイ)	ツクモ特価¥ 69,000
CZ-615D(15型カラーディスプレイテレビ)	ツクモ特価¥132,000
CZ-621D(21型カラーディスプレイ)	ツクモ特価¥125,000

覚えることや整理すること・・・みんなおまかせ!!
PI-3000　定価¥65,000
ZAURUS ザウルス　**ツクモ特価¥49,800**
話題沸騰!情報ツール液晶ペンコム

●MIDIコンピュータミュージック特選セット●

Rolandセット A
- SC-55mkII　¥69,000
- SX-68MII　¥19,800
- Mu-1GS　¥28,000
- **ツクモ特価¥92,000**

Rolandセット B
- CM-500　¥115,000
- SX-68MII　¥ 19,800
- Mu-1GS　¥ 28,000
- **ツクモ特価¥135,000**

KORGセット A
- AG-10　¥49,000
- SX-68MII　¥19,800
- Mu-1GS　¥28,000
- **ツクモ特価¥82,000**

KORGセット B
- 05R/W　¥69,000
- SX-68MII　¥19,800
- Mu-1GS　¥28,000
- **ツクモ特価¥92,000**

X68000/030シリーズ用RAMボード　※価格はすべてツクモ特価

SH-6BE1-1ME(CZ-600C専用)	¥11,000
PIO-6BE1-AE(ACE/PRO/PRO2シリーズ用)	¥11,000
PIO-6BE2-2ME(拡張スロット用)	¥23,000
PIO-6BE4-4ME(拡張スロット用)	¥39,000
SH-5BE4-8M(X68030シリーズ用)	¥46,800
CZ-5BE4(X68030シリーズ用)	¥41,800
CZ-5ME4(CZ-5BE4用拡張RAM)	¥38,000
CZ-6BE2A(XVI専用)	¥42,500
CZ-6BE2D(CompactXVI専用)	¥29,800
TS-6BE2B(CZ-6BE2A/D用拡張RAM)	¥29,800

ソフトウェア

OS-9/X68030 V2.4.5	¥20,000	SOUND SX-68K	¥12,600
Technical Tool Kit V.2.4.5	¥16,000	Matier Ver2.0	¥29,800
UltraC&Professional Pack V1.1	¥36,000	CD-ROM Driver	¥ 4,800
X Windows V11.5	¥24,000	SX-PhotoGallery	¥15,800
SX-WINDOW Ver3.0システムキット	¥15,800	DoubleBookin'	お問い合わせください
SX-WINDOWデスクアクセサリ集	¥11,800	EG Word SX-68K	お問い合わせください
C COMPILER PRO-68K Ver2.1NEWKIT	¥35,800	Workrcom SX-68K(SX-WINDOW開発キット)	お問い合わせください
Easydraw SX-68K	¥15,800	開発キット用ツール集	お問い合わせください
Easypaint SX-68K	¥10,200	倉庫番リベンジSX-68K	¥ 5,440

ツクモIN東京

営 平日 AM10:45～PM7:30 日・祝 AM10:15～PM7:00※東京各店は2月16、17日を臨時休業させていただきます。

ツクモパソコン本店II 4F

担当 荒井
TEL03 (3253) 1899 (直通)
ツクモパソコン本店II代表
TEL03 (3253) 4199 休毎週木曜日

ツクモニューセンター店

担当 沢栄
TEL03 (3251) 0987
休毎週木曜日
※下取り交換、中古販売も行っております。

各店、定休日が祝日と重なる場合は営業致します。

Let's say Motorola っすよ！。JASTのX68kペリフェラル。

■どーも、1か月ぶりのご無沙汰でした。さて、Jリーグの初代チャンピオンはヴェルディ川崎で決着しました。川崎市に住民税を払う隠れアントラーズファンの私としては複雑な心境の限りです。テレビではロス地震の速報やっているし、やっぱり世紀末なんでしょうか。そんな広告担当の個人的話題はさておき、立て続けに発表してまいりましたジャストのX68k周辺機器群、ここまでまとめてご案内いたしましょう。

▽MPUアクセラレーター H.A.R.P for MC68000

型番：DCMA00D1 対応機種：X68000初代,ACE,EXPERT,PRO,SUPER
定価：¥29,800(税別)

■既存のMPUチップと交換するだけであなたのX68kが高速化、そんなおいし過ぎる話しが現実になった弊社のPMUアクセラレーターことH,A,R,P for MC68000。製品発表以来、数多くのX68kユーザーの皆様からの反響と支持を頂き、弊社スタッフ一同、感涙にむせぶ日々が続いております(表現誇張率70％)。X68k究極のボトルネックと呼ばれ続けたその演算速度、ジャスト(他1社)が解決してみせます(笑)。

▽拡張I/Oスロット ESX68

型番：ESX68L4 対応機種：X680x0全機種 定価：¥39,800(税別)

■さて、X68kインフラ整備計画の核、拡張I/OスロットESX68です。マンハッタン野郎の悩みを一挙に解決すべく、高速バッファ搭載のインターフェースカードと拡張I/Oスロット専用電源による徹底した安定性確保、さらに弊社内でも動揺が起きる程のプライスタグで、あなたの財布に鋭く迫ります(笑)。この装備でこの価格、あなたに選択する権利などありません。これを買えば救われるのです。(壺ぢゃーねーって)。

※Motorolaは、モトローラ社の登録商標です。

次回予告　製品ラインアップが一段落し、世界征服への基礎を固めたジャスト他1社、その勢いとは裏腹に、己の意見を記事冒頭に持ってくるわがままな広告担当や、アサヒの烏龍茶しか飲まなくなった驕れる開発スタッフ、この統率を欠いているとしか思えない状況下、あるDSPチップを使ったプロジェクトが進行していた。その名は"OTTO"…。彼らの新たな野望とは、がんばれ高橋市長！、負けるな五十里町長！(それって誰？)。

▽拡張SIMMメモリーボード ER10S

型番：ER10SOn(SIMM未実装) 定価：¥14,800(税別)
：ER10SDn(SIMM4MB1枚実装済) 定価：¥39,800(税別)
対応機種：X68000全機種 '94年3月出荷開始：予約発売中

■安いSIMMが使いたい、手軽にRAMフル実装を実現したい、そんなあなたの切実な願いに応えた、拡張スロットタイプのメモリーボードです。あなたがチョイスするIBM用72ピンSIMMでワンボード10MB実装可能、加えて独自のメモリーサイクルを取ることにより、拡張スロットでも高速なメモリーアクセスが実現できます。SIMMの入手しにくい方の為に実装モデルもリリースする用意周到さ、やっぱりあなたに選ぶ権利はないようです(笑)。

▽MPUアクセラレーター H.A.R.P-FX

型番：DCMA30F1 対応機種：X68030全機種
予価：¥98,000(税別) '94年4月出荷予定：予約受付中

■ついに出たMC68030アクセラレーター、X68030はおろか将来他の030マシンにもターゲットを向けたジャストのフラッグシップです。MPU交換タイプで、ボード上にはMC68030RC50を搭載、030の50MHzあーんどオンチップキャッシュの威力で、貴方はCISCアーキテクチャの限界性能を体験することになるでしょう(誇大な表現かも…)。

■おかげさまで、弱小システムハウス？の弊社にとっては見込み違いのオーダーを皆様からいただく結果となりました。製品出荷が受注に追いつかない、いわゆる「うれしい悲鳴」状態で、注文を頂いたユーザーの皆様にはご迷惑をお掛けしております。今しばらくお待ち下さいますようお願い申し上げます。本当にごめんなさい。もうひとつ、先月アナウンスしていたサポートBBSのご案内です。こちらもひとつよろしくお願いします。

◇JAST's user support BBS 03-3706-7134
(ITU-T V.32bis,V.42bis/LAPM：24Hrs)

サポート (有)エヌ・エム・アイ　開発・販売 (株)ジャスト　JAST
〒156 東京都世田谷区宮坂3-10-7 YMTビル3F TEL.03-3706-9756 FAX.03-3706-9761

いつでもどこでもソフトバンクの14大雑誌

Oh!PC 毎月1.15日発売 定価560円

Oh!X 毎月18日発売 定価600円

Oh!FM TOWNS 毎月18日発売 定価620円

MacUser 毎月18日発売 定価980円

月刊PC 毎月18日発売 定価650円

C MAGAZINE 毎月18日発売 定価980円

DOS/V magazine 毎月8日発売 定価780円

THE WINDOWS 毎月8日発売 定価980円

LAN TIMES 毎月8日発売 定価1,480円

UNIX USER 毎月8日発売 定価980円

PCWEEK 毎週金曜日発売 年間購読料9,000円

月刊情報処理試験 毎月8日発売 定価780円

BEEP! POWERFUL MEGA-MAGAZINE MEGADRIVE 毎月8日発売 定価490円

The スーパーファミコン 隔週金曜日発売 定価380円

SOFT BANK ソフトバンク株式会社 出版事業部

定価はすべて税込です。お近くの書店でお求めください。

5年保証5年保証5年保証5年保証5年保証5年保証5年保証5年保証5年保証

新品パソコン業界最長**5年**保証

全国通販

簡単申込!! 代金引換OK

●店頭販売もしております。御来店大歓迎。
●単品販売もしております。詳しくは、お電話にてお問合せ下さい。
●HARD & SOFT。パソコン専門販売!!

●営業時間:AM10:00〜PM7:00 日・祭:AM10:00〜PM6:00 ●定休日:毎週水曜日・第3日曜日

お問い合わせ お申し込みは フリーダイヤル **0120-01-4865**

お問合せ・お見積 **03-3655-4454** FAX **03-3655-4436**

《業界No.1のメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証。(※モニター・プリンター3年間保証!! ※一部商品は除きます。) ②中古パソコンの1年間保証(※モニター・プリンター6ヶ月間保証!!) ③初期不良交換期間3ヶ月(※新品商品に限らせていただきます。) ④永久買取保証。⑤配達日の指定OK!!(土曜・日曜・祭日もOK) ⑥夜間配達OK!!(※PM6:00〜PM8:00の間。※一部地域は除きます。)

便利でお得な支払いシステム

①翌月一括払い手数料無料(ご利用下さい)。②業界No.1の低金利!! ③月々の支払いは¥1,000より。④9ヶ月先からのスキップ払いOK!! ⑤84回までの分割、ボーナス併用OK!! ⑥カレッジクレジット。⑦ステップアップクレジット。⑧ボーナスだけで10回払いOK!! ⑨現金一括支払いOK!! ⑩商品到着払いOK!!(代引き手数料が必要になります。)(10万円まで900円) ※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。
※店頭即決クレジットにて持ち帰りOK!

注目!!夏のボーナス一括払い、手数料(金利)無料。(平成6年3月末/4月末/5月末/6月末/7月末のいずれかをご指定ください。)

パソコンSHOP P-メディア 春の大特価セール!!

X68030ハードディスクセット

①
330M 外付

●CZ-500C(本体)
●CZ-608D(モニター)
●CS-H300(Filo、12ms)(ターミネーター、ケーブル付)
●フロッピーアタッシュケース(X68ロゴ入り)

合計定価¥572,600
PM特価¥364,000

② 520M 外付

●CZ-500C(本体)
●CZ-608D(モニター)
●CS-H500(Filo、12ms)(ターミネーター、ケーブル付)
●フロッピーアタッシュケース

合計定価¥640,800
PM特価¥395,000

③
80M 内蔵

●CZ-500C(本体)
●CZ-608D(モニター)
●CZ-5H08(シャープ)
●フロッピーアタッシュケース

合計定価¥590,800
PM特価¥370,000

④ 160M 内蔵

●CZ-500C(本体)
●CZ-608D(モニター)
●CZ-5H16(シャープ)
●フロッピーアタッシュケース

合計定価¥627,800
PM特価¥403,000

本体の変更の場合

①CZ-510Cに変更の場合¥71,000
②CZ-300Cに変更の場合¥ 1,000
③CZ-310Cに変更の場合¥64,000
上記加算して下さい。

モニターの変更の場合

①CZ-607D(チューナー付)に変更の場合¥ 3,000
②CZ-614D(チューナー付)に変更の場合¥31,000
③CZ-621D(B) に変更の場合¥58,000
加算して下さい。
※300シリーズにチューナー付のモニターを接続の場合CRTケーブルを購入して下さい。

X68000 Compact XVI セット

①
●CZ-674C(本体)
●CZ-608D(モニター)
●CZ-8NJ2(サイバーステック)

合計定価¥416,600
PM特価¥164,000

②
●CZ-674C(本体)
●CZ-608D(モニター)
●5"FDD(5"×2)(CZ-6FD5同等品)
●CZ-8NJ2(サイバーステック)

合計定価¥456,400
PM特価¥209,000

③
●CZ-674C(本体)
●CZ-614D(ケーブル付)
●CZ-8NJ2(サイバーステック)

合計定価¥466,800
PM特価¥206,000

④

●CZ-674C(本体)
●CZ-614D(ケーブル付)
●5"FDD(5"×2)(CZ-6FD5同等品)
●CZ-8NJ2(サイバーステック)

合計定価¥496,600
PM特価¥246,000

■モニターの変更の場合
CZ-607D-TN(定価¥ 99,800)に変更の場合¥ 3,000
CZ-621D(B)(定価¥168,000)に変更の場合¥58,000
加算して下さい。

X68000/X68030周辺機器

48ドット熱転写カラー漢字プリンタ

■CZ-8PC5-BK
定価¥96,800
特価¥38,000

カラーイメージスキャナ

■JX-325X
定価¥190,000
特価¥143,000

FAX/DATAモデム(マイクロコア)

■MC14400FX
定価¥46,800
特価¥34,500

●CZ-6FD5……………定価¥ 99,800➡特価¥ 49,800
●CS-M120WA(コパル)…定価¥178,000➡特価¥112,000
●LMO-FMX330…………定価¥178,000➡特価¥135,000
●JX-220X……………定価¥168,000➡特価¥ 89,800
●CZ-6TU……………定価¥ 33,100➡特価¥ 23,900
●SH-5BE4-8M(I/O)……………………特価¥ 45,500
●2MB増設RAMボード(I/O)………………特価¥ 23,000
●4MB増設RAMボード(I/O)………………特価¥ 38,300
●CZ-6BE2A……………定価¥ 59,800➡特価¥ 42,800
●CZ-6BE2B……………定価¥ 54,800➡特価¥ 39,300
●CZ-6BE2D……………定価¥ 54,800➡特価¥ 39,300
●CZ-6BM1A……………定価¥26,800➡特価¥19,300
●CZ-6BV1……………定価¥21,000➡特価¥15,200
●CZ-6BS1……………定価¥29,800➡特価¥21,500
●CZ-5MPI……………定価¥54,800➡特価¥42,000
●SX-68MII(MIDI)(サコム)定価¥19,800➡特価¥13,500
●SX-68SC(SCSI)(〃)定価¥26,800➡特価¥17,500
●CZ-5BE4……………定価¥54,800➡特価¥42,000
●CZ-5ME4……………定価¥49,800➡特価¥38,000
●MD-24XT10V(オムロン)‥定価¥29,800➡特価¥22,500
●MD-96XT10V(〃)‥定価¥46,800➡特価¥32,000

■X68000/68030ソフト

CD-ROM Driver Ver.1.06(計測技研)
……定価¥4,800➡**特価¥3,800**
●Z's STAFF PRO68K Ver.3.0(ツアイト) ……定価¥58,000➡特価¥37,500
●たーみのる2(SPS) ……定価¥17,800➡特価¥13,000
●Mu-1 Super(サンワード) ……定価¥39,800➡特価¥28,500
●マチエール Ver.2.0 ……定価¥39,800➡特価¥28,800
●CZ-225BSV Multiword Ver.2.0 ……定価¥32,000➡特価¥23,000
●CZ-247MSD MUSIC PRO68K(MIDI) ……定価¥28,800➡特価¥20,500
●CZ-261MSD MUSIC studio PRO68K Ver.2.0 ……定価¥28,800➡特価¥21,200
●CZ-263GWD Easypaint SX-68K ……定価¥12,800➡特価¥ 9,800
●CZ-264GWD Easydraw SX-68K ……定価¥19,800➡特価¥15,300
●CZ-265HSD New Print Shop Ver.2.0 ……定価¥20,000➡特価¥15,400
●CZ-272CWD Communication SX68K ……定価¥19,800➡特価¥14,500
●CZ-275MWD SOUND SX68K ……定価¥15,800➡特価¥11,500
●CZ-288L WD開発キット(workroom) ……定価¥39,800➡特価¥29,700
●CZ-290TWD SX-WINDOWディスクアクセサリー集 ……定価¥14,800➡特価¥11,500
●CZ-294SS(5")/SSC(3.5")SX-WINDOW Ver.3.0 ……定価¥19,800➡特価¥15,200
●CZ-296LSD C-Compiler PRO68K Ver.2.1 NEW KIT ……定価¥44,800➡特価¥32,500

SHARP 液晶ペンコム PI-3000

定価¥65,000
特価TEL下さい。

●PC-E650
定価¥33,000➡**特価¥25,500**

●PC-E200
定価¥22,000➡**特価¥17,000**

●PC-1262
定価¥24,800➡**特価¥19,000**

特選パソコンラック&OAチェアー
(消費税込み)(送料無料、離島を除く)

4段¥9,785
(W-640)

●ラック&チェアーセット
●布張り(ダークグレー)
●ガスシリンダー
(4段)
¥18,540

■通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。(プリンター・フロッピーの場合、本体使用機種名を明記のこと)
〔銀行振込でお申し込みの方〕●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。(電信扱いでお振込み下さい。)
〔クレジットでお申し込みの方〕●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。●1回〜84回払いまで出来ます。但し、1回のお支払い額は¥1,000円以上。

〔振込先〕 さくら銀行 新小岩支店 普通預金 3384331 (有)ピーメディア

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.9	3.9	4.9	5.4	8.4	11.4	15.9	20.9	26.9	34.9

PM 有限会社 ピーメディア
〒124 東京都葛飾区新小岩2丁目2番地20号
FAX.03-3655-4436
お問い合わせ お申し込みは 0120-01-4865

●本広告の掲載の商品の価格については、消費税は含まれておりません。※掲載の価格は、店頭と異なる場合がございます。※価格は変動することがございますので、最新の価格は、お電話にてお問合わせください。

フリーソフトウェアセレクションVol.2 収録ソフト/データ募集開始!!

～　創造力が、CD-ROMに凝集する　～

X680x0用フリーソフトウェア集CD-ROM、「フリーソフトウェアセレクションVol.2」プロジェクトがいよいよ実働段階に突入しました！

つきましては、本CD-ROMに収録するフリーソフトウェアを募集いたします。プログラム、グラフィックデータ、音楽データ、文書ファイル、なんでも構いません。ただし、他人の著作権を侵害するものはご遠慮ください。みなさんがつちかってきたX680x0文化が、CD-ROMに凝集します。

詳しい募集要綱、および応募フォーマットは、主要パソコン通信ネットワークの掲示板等でご覧になれます。ネットワーカー以外の方は、当社まで電話やFAXにてご請求ください。

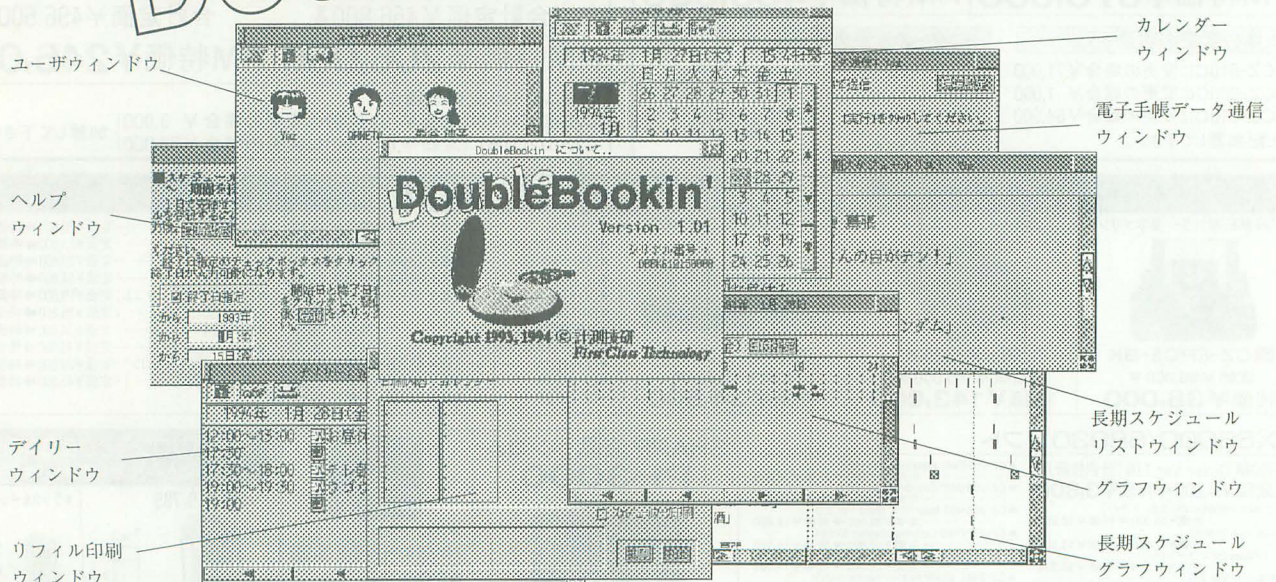
募集要綱の請求、募集に関するお問い合わせは、当社技術部フリーソフトウェアセレクション担当まで。
TEL(0286)38-0301 FAX(0286)38-0305

Vol.1は現在好評発売中です。

発売中　X680x0用フリーソフトウェア集CD-ROM
FreeSoftwareSelection Vol.1　標準価格 ¥5,000

好評発売中!　SX-WINDOW用スケジュール管理ソフト

DoubleBookin'

標準価格 ¥12,800

発売中　CD-ROM Driver Ver1.06　標準価格 ¥4,800

発売中　SX-WINDOW用Photo-CDビュアー
SX-PhotoGallery　基本セット ¥15,000

お求めはお近くのパソコンショップ、または弊社通販部(TEL:0286-22-9811)へお申し込みください。
通販ご希望の方は、ソフト代金+送料￥1,000に消費税を加え、ご住所・お名前・電話番号・商品名を明記した紙を同封の上、現金封筒でお申し込みください。

※記載されている会社名および商品名は各社の登録商標もしくは商標です。

低金利クレジット　通信販売送料 全国一律￥1,000 長期クレジット可能　※表示価格に消費税は含まれておりません
株式会社 計測技研　マイコンショップ BASIC HOUSE 本社／ショールーム／通販部
〒321 栃木県宇都宮市竹林町503-1 TEL 0286-22-9811 FAX 0286-25-3970

CD-ROM Driver情報／CD-ROM Driverは近々バージョンアップしてさらに高機能になる模様。ユーザー登録がまだの方はお早めに。

T1002179030801　雑誌 02179-3

3月号 1994年3月1日発行（毎月1回1日発行）通巻143号 昭和58年11月2日第三種郵便物認可 ソフトバンク株式会社発行 Printed in Japan 特別定価800円（本体777円）